SANDIA REPORT

SAND98–1361
Unlimited Release
Printed June 1998

PRONTO3D User's Instructions:

A Transient Dynamic Code for Nonlinear Structural Analysis

S. W. Attaway, F. J. Mello, M. W. Heinstein, J. W. Swegle, J. A. Ratner, R. I. Zadoks

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND98-1361
Unlimited Release
Printed June 1998

PRONTO3D Users' Instructions: A Transient Dynamic Code for Nonlinear Structural Analysis

S.W. Attaway
Computational Mechanics and Visualization

F.J. Mello
Solid and Structural Mechanics

M.W. Heinstein
Engineering Mechanics and Material Modeling

J.W. Swegle
Experimental Impact Physics

J.A. Ratner
Statistics and Human Factors
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-0443

R.I. Zadoks
Mechanical and Industrial Engineering
The University of Texas at El Paso
El Paso, Texas 79968

Abstract

This report provides an updated set of users' instructions for PRONTO3D. PRONTO3D is a three-dimensional, transient, solid dynamics code for analyzing large deformations of highly nonlinear materials subjected to extremely high strain rates. This Lagrangian finite element program uses an explicit time integration operator to integrate the equations of motion. Eight-node, uniform strain, hexahedral elements and four-node, quadrilateral, uniform strain shells are used in the finite element formulation. An adaptive time step control algorithm is used to improve stability and performance in plasticity problems. Hourglass distortions can be eliminated without disturbing the finite element solution using either the Flanagan-Belytschko hourglass control scheme or an assumed strain hourglass control scheme. All constitutive models in PRONTO3D are cast in an unrotated configuration defined using the rotation determined from the polar decomposition of the deformation gradient. A robust contact algorithm allows for the impact and interaction of deforming contact surfaces of quite general geometry. The Smooth Particle Hydrodynamics method has been embedded into PRONTO3D using the contact algorithm to couple it with the finite element method.

Table of Contents

Introduction

PRONTO3D is a finite element program for the analysis of the three-dimensional response of solid bodies subjected to transient dynamic loading. The program includes nonlinear constitutive models and accurately analyzes large deformations that may lead to geometric nonlinearities. PRONTO3D is a powerful tool for analyzing a wide variety of problems, including classes of problems in:

- impact dynamics,
- rock blasting, and
- accident analysis.

PRONTO3D is a direct descendant of the PRONTO2D code [Taylor, L.M. and Flanagan, D.P., 1987]. Experienced users will recognize the similarity in the structure between PRONTO2D and PRONTO3D, since the theory and algorithms are the same in both codes.

A flexible, problem-oriented language has been developed for the input to PRONTO3D that allows the user to define a complex mechanics problem with a few concise commands. The users' instructions are similar in PRONTO2D and PRONTO3D.

Both on-line and paper versions of the users' instructions are available. The paper version is organized in three sections: introduction and general information, command reference pages, and illustrative examples. The on-line version uses hypertext links to interweave identical parts of the manual.

The development of PRONTO3D was motivated by the need for a code that could serve as a test-bed for research into numerical algorithms and new constitutive models for nonlinear materials. Towards this goal, the code contains a well-documented and easy-to-use interface for implementing new constitutive models. Where possible the element variable names and coding styles are consistent. Comments throughout the code are provided to help developers modify the code for special applications. A developers' guide [Taylor, L.M. and Flanagan, D.P., 1989] further documents the code architecture and individual routines.

PRONTO3D contains no mesh generation or postprocessing capabilities; it relies on external mesh generators and external postprocessors. There are few references to finite element node or element numbers in the problem definition. The philosophy has been to define the problem geometry through the GENESIS mesh definition database [Taylor, L.M., Flanagan, D.P. and Mills-Curran, W.C., 1986].

All boundary conditions are specified through the concept of node and element side sets, which are defined using the GENESIS mesh definition database. The GENESIS database is a subset of the EXODUS [Mills-Curran, W.C., 1988] finite element database. All postprocessing of the finite element results is accomplished by accessing the EXODUS database written by PRONTO3D during the analysis.

PRONTO3D is written in standard FORTRAN [American National Standards Institute, 1978] with some calls to standard C [Kerighan, B.W., and Ritchie, D.M., 1978]. Any system-dependent coding, such as the determination of the date or the memory management, is isolated in libraries, such as SUPES [Flanagan, D.P., Mills-Curran, W.C., and Taylor, L.M., 1986] and SUPLIB. Many of the routines in PRONTO3D are shared through a common source library with PRONTO2D and JAS3D.

Code Overview

Standard File Formats

As a member of the Sandia National Laboratories Engineering Analysis Code Access System (SEACAS) [Sjaardema, G.D., 1993], PRONTO3D benefits from a rich computational analysis environment. Geometry and mesh information for the analysis is read from a file in the GENESIS format [Taylor, L.M., Flanagan, D.P. and Mills-Curran, W.C., 1986], which can be produced by a number of mesh generators and other preprocessors. Results and restart information are written to a file in the related EXODUS format [Mills-Curran, W.C., 1988], which is compatible with a suite of postprocessors and visualization aids.

Four types of output files are available from PRONTO3D: general output (.o); plot (.e); history (.h); and restart (.rsout). The output (.o) file produces a text summary of the problem definition that includes an echo of the input commands, useful derived constants, and any ERROR messages that might be generated from the code. The plot (.e) and history (.h) files are both in the EXODUS format. The plot file stores user requested results for each element at the user's specified times. The history file will store requested results for a select few nodes and elements for every time step. The Restart file (.rsout) is in the same format as the plot file; however, it contains all of the nodal and elemental variables needed to restart the problem. A restart output file can be renamed to create a restart input file (.rsin). To see how to specify these file names see the section PRONTO on UNIX.

Related PRONTO3D Commands

Output Time	Plot Time	Plot Nodal
Plot Element	Plot State	Plot History

Element and Nodal Variables

PRONTO3D defines all of the internal element and nodal variables in such a way that they are available for plotting, restart, element death, or interfacing to other codes. Two types of variables are defined: elemental variables and nodal variables. Variables can be defined as tensors or vectors with the components defined by adding extensions to the variable names (e.g., SIGXX or VELX). For elements with more than one integration point, the station number is also appended to the component name. The elemental variables active during an analysis can change according to the problem definition. To get a list of the variables defined for a given problem, the user can use the Print Info command. Also refer to the lists in Table 1: Nodal and Element Variable Names and Table 2: Material Model State Variable Names.

Related PRONTO3D Commands

Plot Time	Plot Nodal	Plot Element
Plot State	Plot History	Print Info

Element Birth and Death

PRONTO3D has the capability to add elements (element birth) and delete elements (element death) at user selected times in the solution. This can be useful for omitting part of the mesh until it is needed in the calculation. An adaptive element deletion scheme is available to remove any element from the problem, based on the value of any elemental variable. This technique can be used to remove damaged elements from a problem to simulate tearing or fracture. The results from a calculation using element death will be very mesh size dependent.

Related PRONTO3D Commands

Delete Material	Death

Example

Shell Tearing 

Choice of Material Models

At the present time, several nonlinear constitutive material models are incorporated in the program. They include models capable of strain-rate dependent behavior; plasticity models; damage models; hydrodynamic equations of state; soil, foam and concrete models; and explosive burn models. See the Material command to get a list of material models.

The material models in PRONTO3D are all cast in an unrotated configuration, with the rigid body rotations removed from the element strains before the constitutive relation is applied.

Related PRONTO3D Commands

Material	Equation of State

Initial Values

Each material may be assigned an initial value for each component of stress in the reference configuration. The user may also specify a linear variation of stress in the z-coordinate direction. Initial stresses are typically specified to be in equilibrium with the initial boundary conditions. For problems where the initial stresses are complex, the stress field and initial displacements can be read from a restart file. PRONTO3D can read a restart file from the quasi-static code JAS3D.

Related PRONTO3D Commands

Initial Value

Element Types

PRONTO3D currently has Hex, Shell, Spring, and Damper finite elements available for modeling. Beam and Truss elements are planned for the next code release. The eight-noded, Flanagan-Belytschko (F.B.), hexahedron element and the eight-noded, assumed strain, hexahedron element both use single point integration with hourglass control to give an efficient and fast numerical formulation. For fast shell performance, PRONTO3D uses the four-noded Belytschko-Tsay shell.

Rigid bodies can be simulated using either the Hex or the Shell formulation by specifying a material type of Rigid. Springs and Dampers can be used to connect rigid bodies.

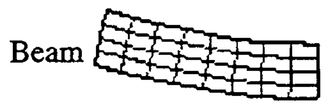
PRONTO3D also can treat problems with the Smooth Particle Hydrodynamics method (SPH). SPH elements are defined as Sphere elements within the GENESIS data file and use the same constitutive relations as the Hex finite elements. SPH elements can be coupled to the finite elements through the finite element contact algorithm.

Eight-Node, Uniform Strain, Hexahedron Element with Flanagan-Belytschko Hourglass Control

PRONTO3D uses the eight-node, three-dimensional, isoparametric element, which is widely used in computational mechanics. A one-point integration of the element underintegrates the element, resulting in a rank deficiency that manifests itself in spurious zero energy modes, commonly referred to as hourglass modes. A two-by-two-by-two integration of the element overintegrates the element and can lead to problems of element locking in fully plastic and incompressible problems. An eight-point integration also carries a tremendous computational penalty compared to the one-point rule. The current formulation eliminates the spurious modes using the hourglass control developed in [Flanagan, D.P. and Belytschko, T., 1981].

For the F. B. hourglass control, the Hourglass Stiffening command can be used to set both the hourglass stiffness and the hourglass viscosity. Experience has shown that both hourglass stiffness and hourglass viscosity are needed to stabilize the eight-node hexahedron element.

Example



Eight-Node, Uniform Strain, Hexahedron Element with Assumed Strain Hourglass Control

An eight-node, assumed strain element formulation is also implemented in PRONTO3D with the Assumed Strain Hourglass command. The formulation is based on [Belytschko, T. and Bindelman, L.P., 1993]. This formulation is both robust and accurate. The stabilization forces are generated by a 2x2x2 Gauss integration of the assumed strain field. The strains are integrated at these eight points; however, the element only evaluates the constitutive model at a single integration point. This element does not require any user-set parameters. The element does not exhibit volumetric locking for incompressible materials and works well for coarse mesh solutions.

For elastic problems, the strain field is constructed so that it represents the beam-bending solution. This allows for accurate elastic solutions for beam-bending problems with only one element through the thickness. For problems where the material behavior becomes plastic, the entire element behaves the same as the center of the element. Thus, for elastic-plastic problems, many elements are still required to accurately track plastic strains.

Two assumptions that are made for this element are: i) the spin is constant within the element, and ii) the material response is constant across the element.

Even with this element, the element aspect ratio should be keep as small as possible to minimize errors.

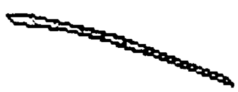
Four-Node, Uniform Strain, Quadrilateral Shell Element

The shell element in PRONTO3D uses a four-node, quadrilateral based on the Belytschko-Tsay formulation [Belytschko, T., Lin, J.I. and Tsay, C.S., 1984]. The element is a four-node quadrilateral with single-point integration in the surface of the element and hourglass control of spurious modes. The thickness direction can be integrated using a variety of different integration schemes. The element formulation uses a corotational coordinate system embedded in the element, which leads to linear equations of motion and constitutive relations that are straightforward.

Theoretically, the eight-node hex element could be used to model any structure. However, analysis of thin structures using hex elements can become impractical because of the large number of elements required. Furthermore, for the hex element, the time step size is controlled by the smallest element dimension. This small time step is required to model the through thickness stress transients and can lead to very long run times. The shell element formulation eliminates the through thickness stress transients, allowing much larger integration time steps.

Example

Shell Beam



Layered Shells

Layered shells are simply a collection of individual shell elements that share the same connectivity. Each may have its own material description, thickness, integration rule, and offset from the mid-surface. A layered shell is technically not an element type but rather a construct for collecting shell elements that will be used to simulate a layered structure. Any shell model available in the code is also available as a layered shell. Since each layer has the same degrees of freedom, the plane of the shell normal remains plane.

Membrane Elements

Membrane elements are provided for efficiently modeling thin structures with negligible local bending stiffness. This element is equivalent to the four-noded shell element with only one integration station through the thickness. The user identifies a quadrilateral mesh as being membrane elements by using the Membrane Scale Thickness command or by setting the integration rule to be one point through the thickness.

Springs and Dampers

Currently springs and dampers are available for connecting rigid bodies to one another or to ground. These elements have been used to simulate bolts and other connecting elements by making small portions of the surfaces to be joined rigid, then connecting the two rigid parts. This helps to distribute the spring or damper force over several points in the model.

Note: The critical time step for a simulation can be dictated by these rigid body mechanisms, and the user may need to adjust the time step. Automatic adjustments are planned for future releases.

SPH Elements

Smooth Particle Hydrodynamics (SPH) is a gridless numerical method that is useful for modeling fluids, explosives, and hydrodynamics [Swegle, J.W., Attaway, S.W., Heinstein, M.W., Mello, F.J. and Hicks, D.L., 1993]. See Smooth Particle Hydrodynamics for more details.

Related PRONTO3D Commands

Time Step Scale	Bulk Viscosity	Hourglass Stiffening
Assumed Strain Hourglass	Shell Hourglass	Shell Integration
Scale Shell Thickness	Membrane Scale Thickness	Layered Shell
Point Mass	Point Inertia	Spring
Damper	Rigid Time Step	SPH

Node Sets

Node sets are used to identify related groups of nodes.

The input commands for PRONTO3D seldomly refer to node numbers directly. Instead, the input will refer to a node set ID. Node sets are read by PRONTO3D from the GENESIS data file. Each node set consists of a node set ID, a list of nodes, and a list of nodal distribution factors. The kinematic boundary conditions are almost always applied to a node set.

Related PRONTO3D Commands

No Displacement	No Rotation	Prescribed Velocity
Prescribed Acceleration	Prescribed Force	Initial Velocity Nodeset
Initial Velocity Angular		

Side Sets

Side sets define groups of related element sides or faces.

Side sets are used to reference surfaces within the finite element mesh. Each side set consists of a side set ID, a list of element sides (element number and face number), and a list of nodal distribution factors. For hex elements, the six element faces can be defined with outward pointing normals. For shell elements, two faces can be defined, one on each side of the shell. Side sets are read from the GENESIS data file.

Side sets are used to define surfaces for pressure loads and for contacts. For pressure loads on hex elements, positive pressure will be applied in the direction of the face normal. Care must be taken to define the correct normal pressure loads on shell elements.

For contact surfaces, side sets can be used to identify which surfaces are to be included in the contacts.

Related PRONTO3D Commands

Pressure	Moving Pressure	Silent BC
Rigid Surface	Contact Surface	Contact Material
Contact Data	Contact Level	

Kinematic Constraints

The geometric boundary conditions allow nodes to be rigidly fixed in space and time or to be defined to move in a specified time-dependent manner. This capability allows for realistic modeling of many physical processes. In general, the user specifies a set of nodes in the GENESIS mesh file using subsets of nodes referred to as Node Sets. Each Node Set is given a unique identification number within the GENESIS mesh file. These Node Sets are then referenced within the PRONTO3D input through their identification numbers.

Related PRONTO3D Commands

No Displacement	No Rotation	Prescribed Velocity
Prescribed Acceleration	Initial Velocity Nodeset	Initial Velocity Material
Initial Velocity Angular	Silent BC	

Loads

PRONTO3D has the ability to apply a variety of mechanical time-dependent and/or time-constant loads to a model. These loads can be point loads, surface pressures, or body forces (arising from acceleration or electromagnetic fields). With these definitions, a great variety of mechanical loading applications can be modeled. In general, pressure loads are defined in the mesh file through subsets of surfaces referred to as Side Sets. Pressures are applied to these surfaces by referring to their identification numbers within the PRONTO3D input stream. Nodal forces and point loads are applied at nodes defined by Node Sets.

In addition, loads can be read from an external load file generated by a separate analysis. The external load file is formatted in such a way that the load can be interpolated both in time and space.

Related PRONTO3D Commands

Cavity Expansion	Prescribed Force	Pressure
Moving Pressure	Gravity	

Contact Surfaces

PRONTO3D can also model contacting surfaces. The contact surfaces can be fixed together, slide without friction, or slide with friction. They can be allowed to close or open as the solution dictates. This capability allows many physical processes to be realistically modeled.

The original algorithm was limited by requiring the user to define contact surface pairs through side sets. The new global contact algorithm allows for automatic contact definition and detection. By using the global contact algorithm, self contact and eroding contacts are easily modeled. With this global contact algorithm, the program will search for all external element faces and add them to a list of surfaces to be searched for contact.

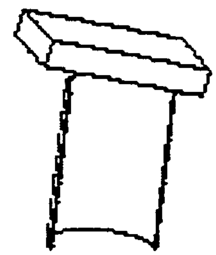
The "fixed" contact surface has also proven useful for grading element size, especially for the three-dimensional problem. This allows for parts of the structure to be very finely modeled to obtain the required resolution. The remainder of the structure, which is required to obtain the global response, can be modeled coarsely. These parts are joined by one or more fixed contact surfaces.

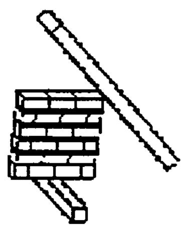
Related PRONTO3D Commands

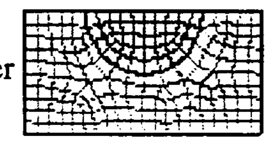
Rigid Surface	Contact Surface	Contact Material
Contact Data	Contact Window	Contact Level

Examples

Cask Impacting Rail 

Can Crush 

Brick Wall 

Contact Chatter 

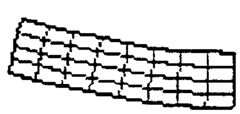
Restart

A capability to stop and restart the solution process is incorporated. The restart can be used to change many of the problem parameters, thus allowing realistic physical processes to be modeled easily. For example, the boundary conditions can be changed at restart. The restart file is written in the EXODUS data format and can be viewed using the same postprocessing tools used on the plot database.

Related PRONTO3D Commands

Termination Time	Read Restart	Write Restart

Example

Beam Restart 

Restart

Smooth Particle Hydrodynamics

Smooth Particle Hydrodynamics (SPH) [Swegle, J.W., Attaway, S.W., Heinstein, M.W., Mello, F.J. and Hicks, D.L., 1993] is a gridless Lagrangian technique that is appealing as a possible alternative to numerical techniques currently used to analyze high-deformation impulsive loading events, such as hypervelocity impact or explosive loading of materials.

SPH has been embedded within PRONTO3D so that each SPH element is a special element type within the finite element architecture. This treatment allows SPH to use the same constitutive models as are used in the finite element method.

SPH differs from standard techniques in that the spatial gradients are approximated using an arbitrary distribution of interpolation points so that no grid is required. At each time step, the nearest SPH neighbors are determined and the velocity gradient and stress divergence are computed based on a kernel sum approximation.

SPH can be coupled to the finite elements through the contact surface algorithm. The ability to couple particle methods and the finite element method allows fluid-structure interaction problems to be solved efficiently.

Related PRONTO3D Commands

SPH
SPH Velocity Smoothing
SPH Variable Smoothing
SPH Symmetry Plane
SPH Viscosity
SPH Interface Smoothing
SPH Kernel Density
SPH Density Normalization
SPH Viscosity Timestep
SPH Decouple Strains
SPH Scale Factor

Rigid Bodies

Any region of a mesh may be treated as a rigid body by specifying the material model to be rigid. The mass center, mass, and inertia for that rigid material block will be calculated and stored. Each time through the time-marching scheme all of the forces applied to nodes belonging to the rigid block are assembled into resultant forces and moments on the rigid body. The rigid body motion is then computed and all of the rigid nodes are placed in their appropriate positions. If two adjacent material blocks are defined as rigid, any shared nodes will move with the second rigid body. They will not act as one rigid body unless they have the same block ID or have been joined by a constraint. The intent of this feature is to allow a detailed model of a structure to be sequentially simplified without remeshing. Portions of a model that only contribute inertially can be modeled with any level of detail, without impacting performance. Rigid bodies can be converted to a flexible material, if necessary, without remeshing. Contacts between rigid bodies and other parts of the model are properly enforced.

Related PRONTO3D Commands

Rigid Body Constraint	Rigid Time Step	Point Mass
Point Inertia	Spring	Damper

Time Step Control

PRONTO3D uses a central difference scheme to integrate the equations of motion through time. The central difference operator is conditionally stable. For hex and shell elements, the code determines the Courant stability limit for each time step using an approximation of the highest eigenvalue in the system. For SPH elements, the stable time step is based on the SPH smoothing size and the material wave speed. For rigid bodies, the time step will usually be controlled by some other aspect of the problem. For the case where only rigid bodies are considered, the user must supply a rigid time step.

When strain softening occurs, the central difference operator is unconditionally unstable. The code allows the time step to be scaled back to a fraction of the initial stable time step for the element when strain softening is detected.

For explosive detonations or for hypervelocity impacts, the initial time step can be set small, then allowed to grow exponentially with time.

Related PRONTO3D Commands

Time Step Scale	Rigid Time Step

Programmed Burn

Explosive events can be simulated in PRONTO3D using what is known as a programmed burn. For a programmed burn, the reaction and initiation of the explosive is not determined by the shock in the material. Instead, the release of chemical energy is determined by the time it takes for a detonation wave to arrive at a material point.

For the programmed burn in PRONTO3D, the pressure is determined by the Jones-Wilkins-Lee (JWL) equation of state. The internal energy in an explosive is initialized by defining the appropriate parameters in the JWL equation of state for the given explosive. After the detonation wave arrives, the internal energy will be converted to a pressure according to the JWL equation of state.

Related PRONTO3D Commands

Detonation Point	Burn Constant

Command Syntax

Keyword Input

Input for PRONTO3D is

- keyword driven, and
- free field format.

The command lines may be in any order. Only the first three characters need to be specified for each word in a command.

Special Characters

- **Comments:** ($) allows the user to place a comment on any line. Anything following a dollar sign on an input line is ignored.
- **Delimiters:** one or more spaces ( ); a comma (,); or an equal sign (=).
- **White space:** any blank space is ignored.
- **Line Continue:** (*) at the end of an input line indicates that the line is continued on the next line. Lines can be 132 characters long.
- **Blank lines:** any blank line is ignored.

Experience has shown that users make fewer errors if the input deck is clear and easy to read. Lots of comments and smart use of white space will help to minimize input errors.

Error Checking

If the code detects an error, it will print an error message that tells the user what is wrong, and then continue to parse the command file. Additional errors would lead to additional error messages. Only after the input is error free will the code run. The code always tries to tell the user why it stopped. Error messages go to the standard output file. PRONTO3D has a standard output (*.o) file, which is the same file that echoes the input and reports progress on execution.

Each input parameter is checked to make sure that its value makes sense. Unless a default is allowed, the code will flag unspecified parameters as errors.

PRONTO on UNIX

General Information

PRONTO3D is supported on the CRAY/SGI, HP, SUN, IBM, DEC, and ASCI Red Unix platforms. The code is run with a script that has the following command format:

```
pronto3d [-help] [-file_options filename] [-options option] [--] [base]
```

The following options are defined:

file_options	Argument	Default	Extension
-input	input_file	fort.5	.i
-output	output_file	fort.6	.o
-mesh	mesh_file	fort.9	.g
-plot	plot_file	fort.11	.e
-rsout	restart_out_file	fort.30	.rsout
-rsin	restart_in_file	fort.32	.rsin
-distributed	dist_load_file	fort.38	.dist
-external	external_file	fort.39	.ext
-thermal	thermal_file	fort.56	.th
-Include	path_to_include	none	(*inc)
-help		NA	(*help)
-MANUAL		NA	(*man)
-executable	executable_file	none	
-aprepro	aprepro_options	none	
-debug		none	(*debug)
-parallel		none	
-subroutine	subroutine_file	none	(*sub)

Notes

- If "base" is specified, then all files not explicitly specified will be read from/written to base.extension.
- (*inc) If -Include=standard is input, then search in:
  `/usr/local/eng_sci/struct/ACCESS/analysis/pronto3d`
- (*help) This option is used to print a usage help file.
- (*man) This option is used to open the users' manual that you are currently reading.

- (*debug) For code to run in the debugger, the source must be compiled and then linked using the debugger.
- (*sub) If the subroutine file ends in ".o", it is assumed to be an object file (already compiled). Otherwise, it will be compiled and linked.

The simplest way to execute the code is to use a base name plus the appropriate suffix for the input, mesh, and other files. In this case, the file default extensions will be assumed, and the command line is simply:

```
pronto3d -- base
```

The file options can be used to specify file names other than the defaults. For information concerning the execution of

Example PRONTO3D Commands

```
pronto3d -- beam
```

The above command will run a problem with the default input files: beam.i, beam.g, and beam.rsin. The default output files will be: beam.o, beam.e, and beam.rsout. The other files may or may not be needed, depending on the nature of the problem. Each of these defaults may be changed using the file options for the PRONTO3D script.

```
pronto3d -exe /scr/username/pronto3d -- beam
```

This command will run the problem, beam, just as in the previous example, but will use the executable image /scr/username/pronto3d.

```
pronto3d -exe /scr/username/pronto3d -debug -- beam
```

This command will run the problem, beam, using the executable image /scr/username/pronto3d within the debugger. This is only useful if the executable has been compiled in debug mode.

```
pronto3d -aprepro -- beam
```

This command will pass the input file beam.i through APREPRO before running PRONTO3D. This is perhaps the most common script option. Omitting the "-aprepro" will cause any APREPRO expressions in the input file to be interpreted as zeros, which usually leads to an instant fatal error.

PRONTO3D Commands

Assumed Strain Hourglass
Bulk Viscosity
Burn Constant
Cavity Expansion
Contact Data
Contact Level
Contact Material
Contact Surface
Contact Exclude
Contact Recompute Concavity
Contact Window
Damper
Death
Delete Material
Detonation Point
Energy Deposition
Equation of State
Exit
Function
Gravity
Hourglass Stiffening
Initial Value
Initial Velocity Angular
Initial Velocity Material
Initial Velocity Nodeset
Layered Shell
Material
Membrane Scale Thickness
Moving Pressure
No Displacement
No Rotation
Output Time
Plot Element
Plot History
Plot Nodal
Plot State
Plot Time
Point Inertia
Point Mass
Prescribed Acceleration
Prescribed Force
Prescribed Velocity
Pressure
Print Info
Read Restart
Rigid Body Constraint
Rigid Surface
Rigid Time Step
Scale Membrane Thickness
Scale Shell Thickness
Shell Hourglass
Shell Integration
Shell Scale Thickness
Silent BC
SPH
SPH Decouple Strains
SPH Density Normalization
SPH Interface Smoothing
SPH Kernel Density
SPH Scale Factor
SPH Symmetry Plane
SPH Variable Smoothing

SPH Velocity Smoothing
SPH Viscosity Timestep
Spring
Subcylcing
Termination Time
Time Step Scale
Title
Write Restart

Title

Command Format

Title

text

Parameters

text	Entire line after the command Title is used as the title.

Description

Use this command to title the analysis. Place the title on the line after the Title command line. The title will be output to the EXODUS data file so that it can be displayed on related graphics.

Example

```
Title
  The title for this analysis
```

Termination Time

Command Format

Termination Time *tend*

Parameters

tend	The time when the analysis should be terminated.

Description

Use this command to set an analysis termination time.

Example

```
Termination Time 100
```

Output Time

Command Format

Output Time *tout*

Parameters

tout	The time interval for printing output. [Default = *tend*/200, where *tend* is defined via the command Termination Time]

Description

Use this command to specify the interval for printed output. Typical output includes the time step number, current solution time, total problem kinetic energy, CPU time per element cycle, and the time dilatation factor. The CPU time per element cycle is the total CPU time for a given time step divided by the total number of elements in the problem. The time dilatation factor can be used to estimate the total CPU time needed for a given problem time. Specifying the command with no value or a value of zero will cause output for each time step.

Example

```
Output Time 50
```

Read Restart

Command Format

Read Restart *restm*

Parameters

restm	The time at which a restart is to begin.

Description

Use this command to identify a restart run. This command restarts the analysis from the restart input file (*.rsin) at the time nearest *restm* (if it is within 5 percent).

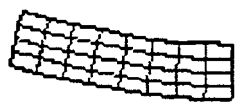
Note: Two different files are used for a restart. *.rsout is written using the Write Restart command. This file must be moved to the *.rsin file before the restart file is read.

Examples

Syntax

```
Read Restart 50
```

Problem

Beam Restart 

Restart

Write Restart

Command Format

Write Restart *trsdmp*

Parameters

trsdmp	The time interval at which to write restart dump files. [Default is to write no restart files]

Description

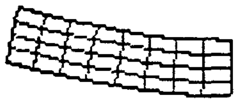
Use this command to write an EXODUS file that can be used as a restart file.

Examples

Syntax

```
Write Restart 50
```

Problem

Beam 

Plot Time

Command Format

Plot Time *tplot, tstart, tend*

Parameters

tplot	Time interval for writing an EXODUS output. [Default = (*tend-tstart*)/10]
tstart	Time to start writing EXODUS output. [Default = 0]
tend	Time to stop writing EXODUS output. [Default = *tend* from Termination Time]

Description

Use this command to specify the frequency with which information is output on the EXODUS plot database. The variables output to the plot database are controlled by the Plot Nodal, Plot Element, and Plot State commands.

Examples

```
Plot Time 50,0,100
```

or

```
Plot Time .001
```

Plot Nodal

Command Format

Plot Nodal *nodal variable 1, nodal variable 2, . . .*

Parameters

nodal variable	Name of the nodal variable to be plotted. [Defaults are displacements, velocities, and accelerations]

Description

Use this command to specify nodal variables to plot. The displacements are always written to the EXODUS file.

Allowable *nodal variable* names are listed in Table 1: Nodal and Element Variable Names. Any request made for a variable name or its alias will result in all components being written. Individual components will appear on the database with the names listed in Table 1 and may be specifically requested by name.

Examples

```
Plot Nodal displacement
```

This command would write the components DISPLX, DISPLY and DISPLZ(3D) to the database.

```
Plot Nodal displx
```

This command would write only the component DISPLX to the output database. This feature is possible due to the memory management scheme in PRONTO3D and may help reduce the size of results files.

Plot Element

Command Format

Plot Element *element variable 1, element variable 2, . . .*

Parameters

element variable	Name of the element variable to be plotted. [Defaults are stresses and energy density]

Description

Use this command to specify element vaiables to plot. Allowable *element variable* names are listed in Table 1: Nodal and Element Variable Names. Any request made for a variable name or its alias will result in all components being written. Individual components will appear on the database with the names listed in Table 1 and may be specifically requested by name.

Examples

```
Plot Element strain
```

This command would write the components EPSXX, EPSYY, EPSZZ(3D), EPSXY, EPSYZ(3D), EPSZX(3D) to the database for all quad(hex) elements and the components EPSXXn, EPSYYn, EPSZZn(3D), EPSXYn, EPSYZn(3D), EPSZXn(3D) for all shell elements (n refers to the integration point).

```
Plot Element epsxx
```

This command would write only the component EPSXX to the output database for all quad(hex) elements, and no shell element strains would be written. Component names must match exactly with a name in Table 1 for data to be written. This feature is possible due to the memory management scheme in PRONTO3D [Taylor, L.M. and Flanagan, D.P., 1989] and may help reduce the size of results files.

```
Plot Element sigxx1, sigxx5
```

This command would write the components of stress at the integration stations 1 and 5 for a shell element.

Plot State

Command Format

Plot State *state variable 1, state variable 2,* **. . .**

Parameters

state variable	Name of the state variable to be written to the EXODUS file. [There are no state variables written to the EXODUS file by default.]

Description

Use this command to specify any of the internal state variables to plot. Table 2: Material Model State Variable Names lists the internal state variable names for each material model. See the PRONTO3D manual [Taylor, L.M. and Flanagan, D.P., 1989] for definitions of these variables.

Note: Under the new memory management employed in PRONTO3D, state variables are stored as components of the element variable SV (see Table 1: Nodal and Element Variable Names), making the Plot State command unnecessary. Any state variable may be requested by name with the Plot Element command. Further, all state variables may be requested by asking for the element variable SV on the Plot Element command. Use the Print Info command to get a listing of the current state variables being used.

Example

```
Plot State  eqps
```

Plot History

Command Format

Plot History *Variable=var name, Coord=*x_0, y_0, z_0, *Name=user name, Comp=comp name, Node=node num, Element=element num*

Parameters

Variable	A keyword that defines the variable to be placed on the database. It can be a nodal, an element, or a state variable name.
var name	Any valid nodal, element, or state variable name or alias. Nodal and element variable names can be found in Table 1: Nodal and Element Variable Names. State variable names are listed in Table 2: Material Model State Variable Names.
Coord	A keyword indicating that the history point will be defined by specifying coordinates.
x_0, y_0, z_0	The coordinates of the history point (two values for two dimensions or three values for three dimensions). If no specific node or element number is provided as part of this command line, PRONTO3D will find the nearest node or element to these coordinates.
Name	A keyword for naming the output variable.

user name A user-defined history output name. This name must be between one and six characters long. If the component specification is omitted, PRONTO3D will construct names for all of the components by using the supplied name and appending the last two characters of the component names listed in Table 1.

Comp An optional keyword used to specify a variable component.

comp name A vector or tensor component specification if used with the *Variable* keyword. Valid values are X, Y, or Z for vectors; XX, YY, ZZ, XY, YZ, or ZX for tensors. If used without the *Variable* keyword, *comp name* is the name of the element variable component (given in Table 1).

Node A keyword indicating a history request at a specific node.

node num A user-supplied node number for which history information is requested.

Element A keyword indicating a history request at a specific element.

element num A user-supplied element number for which history information is requested.

[There are no state variables written to the EXODUS file by default.]

Description

Use this command to specify output to a user-defined history file. The Plot History option differs from the other Plot options in that both a variable and a component may be specified.

This poses a problem for PRONTO3D. Historically only variables which were vectors, symmetric tensors, or state variables could be requested for history plotting. These all have simple components. Vectors have X, Y, and Z; symmetric tensors have XX, YY, ZZ, XY, YZ, and ZX; and state variables do not have any components. It was, therefore, a simple matter to construct the desired component name by concatenating the variable name specified and the given component. This will continue to be the way PRONTO3D tries to decide which variables get written to the database. To allow user access to the data with nonstandard component names, we have changed the input rules. A user may now request a component directly by name.

Minor changes have also occurred in the way history plots of state variables may be requested. The old syntax of specifying the state variable name using the *Variable* keyword still works. As noted above, state variables are now stored internally as components of the element variable SV. Therefore, individual state variables may be requested using the *Comp* keyword, or all state variables may be requested using *Variable*=SV. State variables cannot be requested using both the *Variable* and *Comp* keywords, since concatenation will never produce a valid state variable name.

Examples

Syntax

```
Plot History, VARIABLE=displacement, COMP=X, Node=1,
     Name=Node_1
```

and

```
Plot History, COMP=displx, corrd=0., 0., 0., Name=Node_1
```

will result in displx being written to the history database with the name "DISPLX_NODE_1".

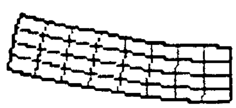
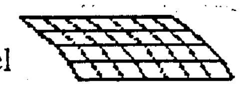
If the user elects to use both the *Variable* and *Comp* keywords, a valid component name must be produced when their arguments are concatenated. For example, the command:

```
Plot History, VARIABLE=rot, COMP=11, Element 1, Name=Mike
```

will cause PRONTO3D to look for a component named rot11, which will not be found. This request is correctly made as:

```
Plot History, COMP=r11, Element 1, Name=Mike
```

Problems

Beam  Shell Cylindrical Panel 

Time Step Scale

Command Format

Time Step Scale *scft, ssft, scinit, scincr*

Parameters

scft — The scale factor to be applied to the internally calculated global time increment.
[Default = 1.0]

ssft — The scale factor to be applied to the internally calculated time step for strain softening elements.
[Default = 0.1]

scinit — The maximum initial time step factor. An initial time step factor of 1.0 means "use the regular initial time step", while a factor of 0.5 means "use half of the

regular initial time step."
[Default = 1.0]

scincr The maximum factor of increase in the time step scale. An increase factor of 1.1 means "increase the time step by no more than 10 percent per cycle," while an increase factor of 1.0 means "no time step increase."
[Default = big number]

Description

Use this command to change the default time step size. If all goes well, you should not have to use this command. See also: Rigid Time Step.

Examples

```
Time Step Scale 0.9
```

Sets the time step size to 0.9 times the internally calculated size.

```
Time Step Scale 0.9 0.5
```

Sets the time step size to 0.9 times the internally calculated size and uses a time step of 0.5 times the initial time step size when strain softening is detected.

```
Time Step Scale 0.9 0.5 0.005 1.1
```

This form of the command is useful for explosive calculations where a very small time step is needed at the beginning of the analysis. The time step is allowed to grow with time.

Bulk Viscosity

Command Format

Bulk Viscosity *b1, b2*

Parameters

b1 The linear bulk viscosity coefficient.
[Default = 0.06]

b2 The quadratic bulk viscosity coefficient.
[Default = 1.2]

Description

Use this command to change default values for the linear and quadratic bulk viscosity coefficients. For most (almost all) calculations, the user should not need to change the default values of the bulk viscosity coefficients.

Example

```
Bulk Viscosity 0.06 1.3
```

Hourglass Stiffening

Command Format

Hourglass Stiffening *hgstiff, hgvis*

Parameters

hgstiff	The hourglass stiffening factor. [Default = 0.05]
hgvis	The hourglass viscosity factor. [Default = 0.0]

Description

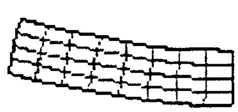
Use this command to change the default values of the hourglass stiffening factor and the hourglass viscosity factor. (A sometimes more accurate hourglass control method is the Assumed Strain Hourglass method.)

Examples

Syntax

```
Hourglass Stiffening .01 .03
```

Problems

Beam 

Cask Impacting Rail 

Assumed Strain Hourglass

Command Format

Assumed Strain Hourglass

Parameters

None

Description

Use this command to have the code use an assumed strain hourglass formulation. For many calculations, this method has been shown to be more accurate and more robust than hourglass stiffness or hourglass viscosity. If Assumed Strain Hourglass is used, then the terms in Hourglass Stiffening are ignored.

Examples

Syntax

```
Assumed Strain Hourglass
```

Problem

Cask Impacting Rail

Shell Hourglass

Command Format

Shell Hourglass *shgmem, shgbnd, shgshr*

Parameters

shgmem The hourglass control parameter for membrane modes.
[Default = 0.03]

shgbnd	The hourglass control parameter for bending modes. [Default = 0.03]
shgshr	The hourglass control parameter for shear modes. [Default = 0.03]

Description

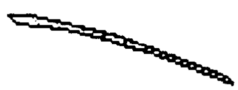
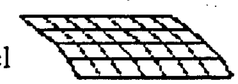
Use this command to change the default hourglass control parameters for shells.

Examples

Syntax

```
Shell Hourglass 0.02 0.01 0.03
```

Problems

Shell Beam  Shell Cylindrical Panel 

Shell Integration

Command Format

Shell Integration *material id, ninteg, rule*

Parameters

material id	This value must match a material id on the GENESIS file.
ninteg	The number of integration points through the thickness. [Default = 5]
rule	Type of integration. Valid integration rules include Gauss, Lobatto, and Trapezoid. [Default = Lobatto]

Description

Use this command to select the number of integration points and the integration rule used through the thickness of shell elements. The table below shows the valid combinations of integration rule and number of integration points.

Valid number of integration points

Rule	ninteg
Gauss	1, 2, 3, 4, 5, 6, 7
Lobatto	3, 4, 5, 6, 7
Trapezoid	1, 2, 3, 4, 5, 6, 7, 8, 9

Examples

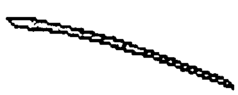
Syntax

```
Shell Integration 100, 3, gauss
```

The above command will set Material 100 to have a three-point Gauss integration rule.

Three integration points will give good results for problems that remain elastic. For problems where plasticity is present, more integration points will give better results.

Problem

Shell Beam 

Scale Shell Thickness

Command Format

Scale Shell Thickness *material id, scale factor*

Parameters

material id	This value must match a material id on the GENESIS file.
scale factor	The scale factor to be applied to the shell thickness. [Default = 1.0]

Description

Use this command to multiply the thickness attribute of a shell element block by a scale factor. The initial thickness attribute is read from the GENESIS data file. This command is the same as Shell Scale Thickness.

Example

```
Scale Shell Thickness 5 0.01
```

Multiply the shell thickness for *material id* 5 by 0.01.

Shell Scale Thickness

Command Format

Shell Scale Thickness *material id, scale factor*

Parameters

material id	This value must match a material id on the GENESIS file.
scale factor	The scale factor to be applied to the shell thickness. [Default = 1.0]

Description

Use this command to multiply the thickness attribute of a shell element block by a scale factor. The initial thickness attribute is read from the GENESIS data file. This command is the same as Scale Shell Thickness.

Example

```
Shell Scale Thickness 5 0.01
```

Multiply the shell thickness for *material id* 5 by 0.01.

Scale Membrane Thickness

Command Format

Scale Membrane Thickness *material id, scale factor*

Layered Shell

Command Format

Layered Shell

material id, offset
material id, offset
...
...
material id, offset
end

Parameters

material id	Must match a material block id on the GENESIS file.
offset	Locates the mid-plane of the layer relative to the plane of the mesh.

Description

Use his command to construct a multilayered shell by meshing the same region multiple times. The individual meshes have distinct material ids and are organized into a layered shell as shown in the example below.

Each layer in a Layered Shell definition must have the same mesh connectivity. The thickness of each layer may be scaled using the Scale Shell Thickness command.

Examples

```
Layered Shell
   1, 5.2
   2, -5.2
end
```

In the above example, two shell layers are constructed from Material 1 and 2. The shells are offset from the neutral axis by 5.2 units.

```
Layered Shell
   1, -.05
   2, 0.0
   3,  .05
end
Scale Shell Thickness, 1, .04
Scale Shell Thickness, 2, .06
Scale Shell Thickness, 3, .04
```

The Layered Shell in the above example would have a set of layers as shown in Figure 1.

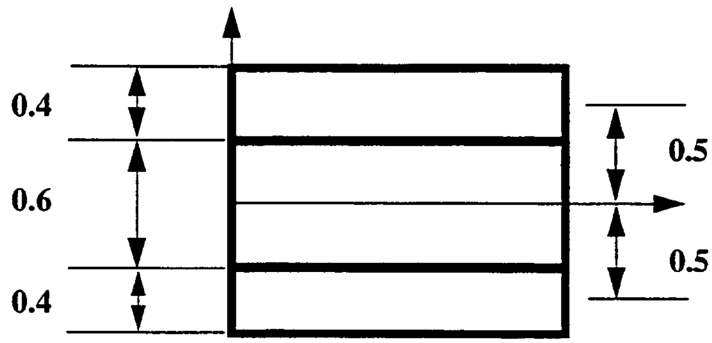


Figure 1 **Layered Shell Example**

Exit

Command Format

Exit

Parameters

None

Description

Use this command to terminate command input. The remaining lines in command file are ignored.

Example

```
Exit
```

Function

Command Format 1

Function *function id, [linear]*

$x_1, f(x_1)$

$x_2, f(x_2)$

...

$x_n, f(x_n)$
end

Command Format 2

Function *function id, polynomial*

a_0
a_1
...
a_n
end

Parameters

function id	Any nonzero integer by which you wish to identify this function. Each function must have a unique id.
linear	Function is a linear interpolation between x, f(x) pairs as listed following the command Function line. [Default is linear]
polynomial	Function is a polynomial, up to order 6, defined by

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \quad (1)$$

Description

Use this command to define linear or polynomial functions. These functions are used in the Prescribed Velocity, Prescribed Acceleration, Prescribed Force, Pressure, and many of the Material model definitions.

After a Function command you must enter a list of points defining the function. For a Linear function, each abscissa-ordinate pair is input on a separate line immediately following the Function command line. As shown in the first example below, the list is terminated by a line containing the END command. The abscissa of a linear function must increase monotonically. PRONTO3D linearly interpolates between function points, but does not extrapolate. If the argument to the function falls outside of the user specified range, PRONTO3D ignores the boundary condition or load associated with that function. This means that a boundary condition can turn on or off at a specific time.

For a Polynomial function, the coefficients of the polynomial must be listed immediately following the Function command line. Each coefficient of the polynomial is input on a separate line. A polynomial of up to 6th order may be defined (i.e., $n \leq 6$). As shown in the second example below, the list is terminated by a line containing the END command.

Example 1

```
Function 100, linear
   0, 0
   1, 100.5
   10, 5012.3
end
```

Example 2

```
Function 200, polynomial
   5
   -4
   20
end
```

For this example, the polynomial function will be computed as

$$f(x) = 5 + (-4)x + 20x^2 \tag{2}$$

Example 3

```
$Example constant polynomial
Function 10, polynomial
   1.
end
```

This example defines a polynomial function that has the constant value of 1.0.

Example 4

```
$ Example simple function.
Function 10
   0.0 0.0
   0.5 2.5
   1.0 10.0
end
```

As shown in Figure 2 below, the simple function example above defines a function valid for $0.0 < x < 1.0$.

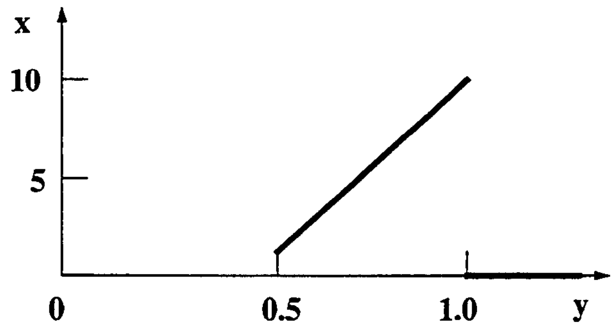


Figure 2 **Function Definition**

Example 5

```
$Example sine function using APREPRO
$ {x = 0} {angle = 10}
Function 100
{loop(angle)}
{x++} {sind(x)}
{endloop}
End
```

The above example shows how to construct a sine function using the loop option in APREPRO. The APREPRO output generated by the previous commands is shown below.

```
Function 100
0 0.01745240644
1 0.0348994967
2 0.05233595624
3 0.06975647374
4 0.08715574275
5 0.1045284633
6 0.1218693434
7 0.139173101
8 0.156434465
9 0.1736481777
End
```

No Displacement

Command Format

No Displacement *direction, node set id*

Parameters

direction	X, Y, or Z.
node set id	This value must match a node set on the GENESIS file.

Description

Use this command to enforce zero displacement in the specified direction for each node in *node set id*.

Example

```
No Displacement X, 1
```

No Rotation

Command Format

No Rotation *direction, node set id*

Parameters

direction	X, Y, or Z.
node set id	This value must match a node set on the GENESIS file.

Description

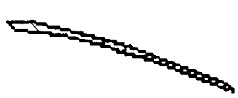
Use this command to enforce zero rotation in the specified direction for each node in *node set id*.

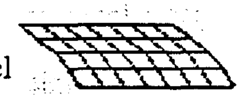
Examples

<u>Syntax</u>

```
No Rotation X, 1
```

<u>Problems</u>

Shell Beam 

Shell Cylindrical Panel 

Prescribed Velocity

Command Format

Prescribed Velocity *direction, node set id, function id, scale factor,* c_x, c_y, c_z, n_x, n_y, n_z

Parameters

direction	X, Y, Z, Radial, Cylindrical, Normal, Spherical, Rotational, or External.
node set id	This value must match a node set on the GENESIS file.
function id	This value must match a function defined via the Function command.
scale factor	Scales the function value. [Default = 1.0]
c_x, c_y, c_z	The center point coordinates. These coordinates are defined only for options Radial, Cylindrical, Spherical or Rotational.
n_x, n_y, n_z	The axis or normal vector. This vector is defined only for options Radial, Cylindrical, Normal, or Rotational.

Description

Use this command to set the appropriate component of velocity of each node in *node set id* to the product of the *function id* value and the *scale factor*. The Radial option defines the radial velocity component with respect to a cylindrical coordinate system defined by the center point and axis vector. The Cylindrical option defines the tangential (counterclockwise) velocity with respect to this cylindrical coordinate system. The Normal option simply defines a Cartesian component in a direction that is not aligned with one of the coordinate axes. The Spherical option defines the radial velocity with respect to a spherical coordinate system. Finally, the Rotational option defines the angular velocity for pure circular motion about the defined axis.

The value of velocity has units of $[\text{distance}] \div [\text{time}]$ for all options **except** the Rotational option. For the Rotational option, the velocity has units of radians $\div [\text{time}]$. The actual velocities of the nodes in *node set id* are determined by multiplying the Prescribed Velocity value by the radial distance (from the center point *c* as projected on the plane normal to the axis *n*).

Examples

```
Prescribed Velocity X, 1, 1, 2

Prescribed Velocity External, 1
```

The External option allows velocities for a set of points to be read from a file. When the External option is specified, the only other input required is the *node set id*. PRONTO3D will look to an external file (.ext) for the following information:

```
numpoints
x_{i=1,n}, y_{i=1,n}, z_{i=1,n}
time1, pressure_{i=1,n}, velx_{i=1,n}, vely_{i=1,n}, velz_{i=1,n}
time2, pressure_{i=1,n}, velx_{i=1,n}, vely_{i=1,n}, velz_{i=1,n}
...
timeN, pressure_{i=1,n}, velx_{i=1,n}, vely_{i=1,n}, velz_{i=1,n}
```

The coordinates for each node in *node set id* list will be matched with the closest coordinate on the external velocity tape. Once all the node set points have been mapped to the points on the tape, the velocity of each point is prescribed using linear interpolation between the times listed on the tape. The pressure field should be set to zero if no prescribed external pressures are specified.

Prescribed Acceleration

Command Format

Prescribed Acceleration *direction, node set id, function id, scale factor,* c_x, c_y, c_z, n_x, n_y, n_z

Parameters

direction	X, Y, Z, Radial, Cylindrical, Normal, Spherical, or Rotational.
node set id	This value must match a node set on the GENESIS file.
function id	This value must match a function defined via the Function command.
scale factor	Scales the function value. [Default = 1.0]
c_x, c_y, c_z	The center point coordinates. These coordinates are defined only for options Radial, Cylindrical, Spherical, or Rotational.
n_x, n_y, n_z	The axis or normal vector. This vector is defined only for options Radial, Cylindrical, Normal, or Rotational.

Description

Use this command to set the appropriate component of acceleration of each node in *node set id* to the product of the *function id* value and the *scale factor*. The Radial option defines the radial acceleration component with respect to a cylindrical coordinate system defined by the center point and axis vector. The Cylindrical option defines the tangential (counterclockwise) acceleration with respect to this cylindrical coordinate system. The Normal option simply defines a Cartesian component in a direction that is not aligned with one of the coordinate axes. The Spherical option defines the radial acceleration with respect to a spherical coordinate system. Finally, the Rotational option defines the angular acceleration for pure circular motion about the defined axis.

The value of accleration has units of $[\text{distance}] \div [\text{time}]^2$ for all options **except** the Rotational option. For the Rotational option, the accleration has units of radians $\div [\text{time}]^2$. The actual acclerations of the nodes in *node set id* are determined by multiplying the Prescribed Acceleration value by the radial distance (from the center point *c* as projected on the plane normal to the axis *n*).

Example

```
Prescribed Acceleration X 10 1 1.0
```

Prescribed Force

Command Format

Prescribed Force *direction, node set id, function id, scale factor,* c_x, c_y, c_z, n_x, n_y, n_z

Parameters

direction	X, Y, Z, Radial, Cylindrical, Normal, or Spherical.
node set id	This value must match a node set on the GENESIS file.
function id	This value must match a function defined via the Function command.
scale factor	Scales the function value. [Default = 1.0]
c_x, c_y, c_z	The center point coordinates. These coordinates are defined only for options Radial, Cylindrical, or Spherical.
n_x, n_y, n_z	The axis or normal vector. This vector is defined only for options Radial, Cylindrical, or Normal.

Description

Use this command to set the appropriate component of force on each node in *node set id* to the product of the *function id* value and the *scale factor*. The Radial option defines the radial force component with respect to a cylindrical coordinate system defined by the center point and axis vector. The Cylindrical option defines the tangential (counterclockwise) force with respect to this cylindrical coordinate system. The Normal option simply defines a Cartesian component in a direction that is not aligned with one of the coordinate axes. Finally, the Spherical option defines the radial force with respect to a spherical coordinate system.

Example 1

```
Prescribed Force X, 1, 1, 2
```

Example 2

```
Prescribed Force Z, 100, 20
Function 20
  0. 1.
  1. 1.
end
```

Initial Velocity Nodeset

Command Format

Initial Velocity Nodeset *node set id*, v_x, v_y, v_z

Parameters

node set id	This value must match a node set on the GENESIS file.
v_x, v_y, v_z	A velocity vector.

Description

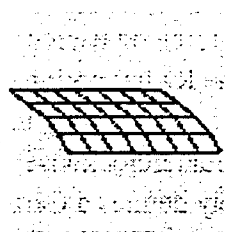
Use this command to initialize each velocity component for each node in *node set id* to the specified value.

Example 1

Syntax

```
Initial Velocity Nodeset 10 5.0 0. 1.0
```

Problem

Shell Cylindrical Panel 

Example 2

```
$ Example Impact Angle using APREPRO
$ {Units('SI')}
${speed = 100*mph}
${angle = 45*deg}
Initial Velocity Nodeset 10 -
{speed*cosd(angle)} -
{speed*sind(angle)} -
{velz= 0.0}
```

This example allows the user to input the impact angle and impact speed as parameters that are later used to compute the x and y components of velocity. It also illustrates how APREPRO can be used to convert the units miles per hour to the units meter per second.

Initial Velocity Material

Command Format

Initial Velocity Material *material id,* v_x, v_y, v_z

Parameters

material id	This value must match an element block on the GENESIS file.
v_x, v_y, v_z	A velocity vector.

Description

Use this command to initialize each component of the velocity of each node connected to the specified *material id* to the specified value.

Example

```
Initial Velocity Material 10 5. 0. 1.0
```

Initial Velocity Angular

Command Format

Initial Velocity Angular *material id,* ω_x, ω_y, ω_z, c_x, c_y, c_z

Parameters

material id	This value must match an element block on the GENESIS file.
ω_x, ω_y, ω_z	The angular velocities in the x, y, and z directions, respectively, in (radians per second).
c_x, c_y, c_z	The center point coordinates. [Default = 0, 0, 0]

Description

Use this command to initialize the velocity of each node connected to the specified *material id* to correspond to the given angular velocity field. The velocity vector for each node is calculated using the cross product of the angular velocity and the position vector from the center point to the node.

Example

```
Initial Velocity Angular 10 5. 0. 1.
```

Pressure

Command Format

Pressure *side set id, function id, scale factor*

Parameters

side set id	This value must match a side set on the GENESIS file.
function id	This value must match a function defined via Function. If the *function id* is replaced by the keyword External, then the pressure will be read from an external file. (.ext)

scale factor	Scales the function value. [Default = 1.0]

Description

Use this command to apply a pressure equal to the product of the *function id* value and *scale factor* to each element side in *side set id*. The calculated pressure value at each side node is multiplied by its side set scale factor as read from the GENESIS file. A positive pressure is directed inward to the brick elements. On the front surface of the shell element $\hat{x} = \frac{h}{2}$, a positive pressure acts in the direction opposite the shell normal. On the back surface of the shell element $\hat{x} = -\frac{h}{2}$, a positive pressure acts in the same direction as the shell normal. The positive direction for the shell normal is described in [Bergmann, V.L., 1991].

The format of the External pressure file is the same as the format described in the Prescribed Velocity command.

Example

```
Pressure 1, 1, 1.0
```

Moving Pressure

Command Format

Moving Pressure *side set id,* c_x, c_y, c_z, *peak id, rise id,* C_p, t_0, *scale factor*

Parameters

side set id	This value must match a side set on the GENESIS file.
c_x, c_y, c_z	The center point coordinates.
peak id	This value must match a function defined via Function.
rise id	This value must match a function defined via Function.
C_p	The propagation speed.
t_0	The arrival time.
scale factor	Scales the peak function value. [Default = 1.0]

Description

Use this command to a moving pressure to a side set. The moving pressure boundary condition implemented in PRONTO3D represents a relatively simple way of incorporating both a spatial and temporal distribution of pressure loading on a surface. The implementation described here is intended for blast-type loading on a surface where the blast originates form some point defined by the coordinates (c_x, c_y, c_z) and propagates along the surface. Assume that the surface is flat and the distance from any point on the surface to point (c_x, c_y, c_z) is given by d. Then the pressure at any point is written as

$$p(\tau, d) = a\tau e^{-b\tau} \tag{3}$$

where τ is the time measured from the arrival of the pressure wave at the point; and a and b are functions of distance which are defined below. If w is the propagation speed of the pressure wave along the surface, then τ is given by

$$\tau = t_0 - \frac{d}{w} \tag{4}$$

where t_0 is the pressure initiation time at the point (c_x, c_y, c_z). The time at which Equation (3) gives a maximum for the pressure is given by

$$\tau_{max} = \frac{1}{b} \tag{5}$$

which we refer to as the rise time. The peak pressure obtained at this time is

$$p_{max} = \frac{a}{b}e^{-1} \tag{6}$$

The user defines two functions of distance from the point (c_x, c_y, c_z) which describe the behavior of the pressure wave. The first function defines the peak pressure as a function of distance, while the second describes the rise time as a function of distance. Using Equation (5) and (6), the parameters a and b as functions of distance are written as:

$$a(d) = \frac{f_1(d)}{f_2(d)}e^{1} \tag{7}$$

$$b(d) = \frac{1}{f_2(d)} \tag{8}$$

The user can define the functions in any manner necessary to allow for a quite general specification of the moving pressure wave. If the user inputs a zero value of the propagation speed, w, the code assumes that the pressure is applied instantaneously along the surface (i.e., this corresponds to an infinite propagation speed).

Example

```
Moving Pressure 100, 0, 0, 0, 10, 20, 500., 1.e-3, 1000.
```

In this example a moving pressure is applied to *side set id* 100, at the center point coordinates (0,0,0). The peak pressure value is defined by the product of 1000. and the value of Function 10, which is evaluated as a function of distance from the center point. The rise time is defined by the product of 1.e-3 and the value of Function 11, which is also evaluated as a function of distance from the center point. The pressure wave is propagated at 500 (units of distance per time unit).

Silent BC

Command Format

Silent BC *side set id*

Parameters

side set id	This value must match a side set on the GENESIS file.

Description

Use this command to set a nonreflecting boundary condition that is applied to each element side in *side set id.*

Example

```
Silent BC 10
```

Rigid Surface

Command Format

Rigid Surface *side set id,* c_x, c_y, c_z, n_x, n_y, n_z, μ

Parameters

side set id	This value must match a side set on the GENESIS file.
c_x, c_y, c_z	The center point coordinates.
n_x, n_y, n_z	The outward normal vector.

μ	The static coefficient of friction. [Default = 0.0]

Description

Use this command to set a rigid surface condition that is enforced for all nodes in *side set id.*

Example

```
Rigid Surface 10 0. 0. 0. 0. 1. 0. 0.2
```

The example above shows how a rigid surface passing through point (0,0,0) with an outward normal in the y-direction would be defined (see Figure 3). All nodes in *side set* 10 will be checked to see if they are penetrating the rigid surface. If penetration occurs, then the nodal force required to push the penetrating nodes back to the rigid surface will be generated. The nodal force can be output to the plotting data file using the Plot Nodal command with the nodal variable React.

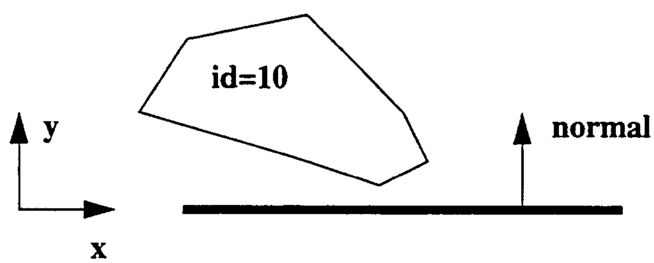


Figure 3 Rigid Surface Example

Rigid Body Constraint

Command Format

Rigid Body Constraint *type, body1, [body2, x, y, z,* (i_x, i_y, i_z) *]*

Parameters

type	The type of constraint.
body1	The material id of the first rigid body in the constraint. (No further data is required for kinematic constraint types.)
body2	The second rigid body in the constraint. (Needed for rigid body joints only.)
x, y, z	The global location of the joint. (Not used for Rigid joint type.)

i_x, i_y, i_z A vector.

Description

Use this command to specify rigid body joints as Spherical, Universal, Revolute, or Rigid. Kinematic constraints are available as DISPLX, DISPLY, DISPLZ, VELX, VELY, VELZ, ACCLX, ACCLY, ACCLZ, ROTX, ROTY, ROTZ, OMEGAX, OMEGAY, OMEGAZ, ALPHAX, ALPHAY, ALPHAZ, SYMMX, SYMMY, or SYMMZ.

For Universal joint types, i_x, i_y, i_z gives the orientation of the `i` axis (Figure 4). If no orientation is given on the input line, the `i` direction will be calculated as the cross product of the vectors from the body centers to the connection. A fatal error will be reported if the connection falls on the line connecting the body centers, and no `i` orientation is given.

For Revolute joints types, i_x, i_y, i_z gives the orientation of the hinge axis. If no orientation is given on the input line, the hinge axis will be calculated as the cross product of the vectors from the body centers to the connection. A fatal error will be reported if the connection falls on the line connecting the body centers, and no `i` orientation is given.

Examples

```
Rigid Body Constraint, spherical, 10, 0, .1, .2, .3
```

This command results in a pinned connection between *material block* 10, which must be a rigid material, and ground. The joint is located at global coordinates (.1, .2, .3) at time 0. Note, the vector i_x, i_y, i_z is not required for this joint type.

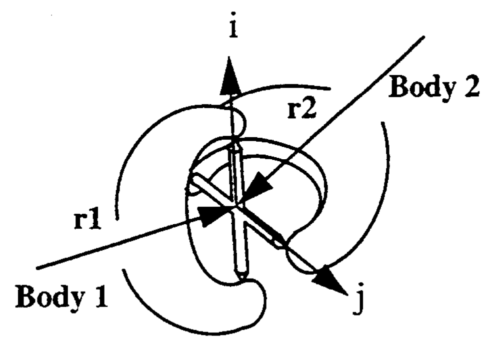


Figure 4 Universal Joint

```
Rigid Body Constraint, universal, 10, 20, .1, .2, .3 , 1., 0.,
    0.
```

This command results in a universal joint connecting *material blocks* 10 and 20, which must be rigid materials. The joint is located at global coordinates (.1, .2, .3) at time 0. Note, the vector i_x, i_y, i_z is used to indicate the axis of the gimbal in body 10 is along the global x direction.

```
Rigid Body Constraint, revolute, 10, 20, .1, .2, .3 , 1., 0.,
    0.
```

This command results in a revolute joint connecting *material blocks* 10 and 20, which must be rigid materials. The joint is located at global coordinates (.1, .2, .3) at time 0. Note, the vector i_x, i_y, i_z is used to indicate the axis of the hinge is along the global x direction.

All Kinematic constraints are prescribed with the syntax:

```
Rigid Body Constraint, displx, 10
```

BIG IMPORTANT NOTE: Rigid bodies ignore all kinematic constraints specified on their nodes. This means you do not have to exclude rigid nodes from a node set that has a kinematic constraint. However, it also means that a rigid body must be given appropriate constraints through this command. This usually arises in a symmetry situation where a node set is established along the symmetry line and given the required constraints. If some of the nodes in the node set are rigid, the rigid body will not know anything about a symmetry boundary unless it is specified with a Rigid Body Constraint command.

Point Mass

Command Format

Point Mass *block id, mass value, x, y, z*

Parameters

block id	This value must match a rigid material block on the GENESIS file.
mass value	The value of the lumped mass.
x, y, z	The global location of the lumped mass.

Description

Use this command to associate a lumped mass with a rigid material block. This is commonly used to represent a discrete component that is not modeled. For example, if an automobile is to be treated as a rigid body, the user may mesh a simplified exterior shape and account for other massive componenents, such as the battery, engine, tires, frame, fuel, etc., by using lumped masses. This is simply a convenient way for the user to better approximate the mass distribution in a rigid body. Note that a point mass does not have to correspond to a point occupied by the rigid body.

Example

```
Point Mass 25 .01 1., 0., 2.
```

Point Inertia

Command Format

Point Inertia *block id, I_1, I_2, I_3, [Global]*

Parameters

block id	This value must match a rigid material block on the GENESIS file.
I_1, I_2, I_3	The three principal inertia values.
Global	An optional keyword that indicates that the principal directions are the global directions.

Description

Use this command to specify principal point inertia values to be associated with a rigid body material block. If *Global* is not specified on the command line, then the three principal directions are input as unit vectors, one per line, followed by an *End* statement, as shown in the second example below. The Point Inertia command, like the Point Mass command, allows the user to add inertia to a rigid body.

In the example in the Point Mass command description, if an automobile is to be modeled as a rigid body, then the engine inertia can be added with a Point Inertia command. The inertia of a component will typically be given or be most easily calculated about principal axes. It is often possible for a CAD pacakage to produce mass, center of mass (c.g.), and inertia information for components and collections of components. This feature in PRONTO3D allows the user to input this information directly, without having to build detailed meshes. Therefore, the complete effect of the engine on the automobile's mass and inertia can be accounted for by giving the engine's mass and center of gravity location, along with its three principal inertias and the principal axes.

Example 1

```
$ first format
$ Point Inertia mid, i1, i2, i3, Global

Point Inertia 1; 100, 200, 30, Global
```

Here the specified inertia values ($I_1 = 100$, $I_2 = 200$, $I_3 = 30$) are input in global coordinates.

Example 2

```
$ Second format
Point Inertia mid, i1, i2, i3
```

```
    x1, y1, z1
    x2, y2, z2
    x3, y3, z3
end
```

Spring

Command Format

Spring *type, body1, body2, function id,* $x_1, y_1, z_1, x_2, y_2, z_2$

Parameters

type	The type of spring.
body1	The block id of the first rigid material attached by the spring.
body2	The block id of the second rigid material attached by the spring. (A value of zero implies that the attachment is between *body1* and ground).
function id	The spring force-deflection relationship which must match a Function definition.
x_1, y_1, z_1	The global location of the attachment to *body1*. (This point need not be interior to *body1*.) Note: these values are ignored for spring type ROT.
x_2, y_2, z_2	The global location of the attachment to *body2*. (This point need not be interior to *body2*.) Note: this location defines the rotation axis, embedded in *body2*, for spring type ROT.

Description

Use this command to define a spring (force versus displacement) element and its position in the mesh. Available spring types are: PPT (point to point total relative displacement); PPX (point-to-point relative x displacement); PPY (point-to-point relative y displacement); PPZ (point-to-point relative z displacement); and ROT (relative rotation).

Example

```
Title
    spring test problem
Termination Time .5005
Plot Time 0.0
Output Time 0.0
Plot History variable=displ component=y node=1 name=nd1
```

```
Rigid Time Step .001
Material 1 rigid .1
   contact modulus 1000.
end
$Spring ppt 1 0 10 0 0 -.5 0 1. -.5
Function 10
   0,0
   1.0 1000.
end
Function 20
   0,0
   .0001 1.0
   10.0 1.0
end
Gravity 20 0 -386.4 0
Exit
```

This example shows a spring that is attached between material 1, which is a rigid body, and the ground point (0.,1.,-.5). Function 10 describes the spring force-displacement relation. The Gravity command is used to load the rigid body and spring.

Damper

Command Format

Damper *type, body1, body2, function id,* x_1, y_1, z_1, x_2, y_2, z_2

Parameters

type — The type of damper.

body1 — The block id of the first rigid material attached by the damper.

body2 — The block id of the second rigid material attached by the damper. (A value of zero implies that the attachment is between *body1* and ground).

function id — The damping force-velocity relationship which must match a Function definition.

x_1, y_1, z_1 — The global location of the attachment to *body1*. (This point need not be interior to *body1*.) Note: these values are ignored for damper type ROT.

x_2, y_2, z_2 — The global location of the attachment to *body2*. (This point need not be interior to *body2*.) Note: this location defines the rotation axis, embedded in *body2*, for damper type ROT.

Description

Use this command to define a damper (force versus velocity) element and its position in the mesh. Available damper types are: PPT (point-to-point total relative velocity); PPX (point-to-point relative x velocity); PPY (point-to-point relative y velocity); PPZ (point-to-point relative z velocity); and ROT (relative rotational velocity).

Example

```
Title
    damper test problem
Termination Time .2001
Plot Time 0.0
Output Time 0.0
Plot History variable=displ component=y node=1 name=nd1
Rigid Time Step .001
Material 1 rigid .1
   contact modulus 1000.
end
Spring ppt 1 0 10 0 0 -.5 0 1. -.5
Function 10
   0,0
   10.0 10000.
end
Damper ppt 1 0 20 0 0 -.5 0 1. -.5
Function 20
   0,40.
   100.0 40.
end
Gravity 30 0 -386.4 0
Function 30
   0,1.0
   10.0 1.0
end
Exit
```

This example attaches a Spring and a Damper to the rigid Material 1 at (0.,0.,-0.5). The other end of the Spring and Damper are attached at the ground point (0. 1. -0.5). The damper force-velocity relationship is defined by Function 20. The Spring and Damper system is loaded by Gravity.

Rigid Time Step

Command Format

Rigid Time Step *value*

Parameters

value The time step for strictly rigid material problems.

Description

Use this command to set the time step for an entirely rigid body problem. Stable time step calculations are based on the minimum transit time of a disturbance across an element. For a rigid body this time is not defined. Currently some trial and error may be required to select an appropriate time step. Automatic rigid time step control is being developed.

Examples

```
Rigid Time Step .001
```

Contact Surface

Command Format 1 (paired contact)

Contact Surface *side 1 id, side 2 id,* μ_0, β, μ_1, γ

Command Format 2 (fixed paired contact)

Contact Surface *side 1 id, side 2 id, FIXED,* β, *toler*

Command Format 3 (global contact)

Contact Surface *side 1 id*

Parameters

side 1 id This value must match a side set on the GENESIS file.

side 2 id This value must match a side set on the GENESIS file.

μ_0 The static coefficient of friction.
[Default = 0.0]

β The kinematic partition factor.
[Default = 0.5]

μ_1 The high velocity coefficient of friction.
[Default = 0.0]

γ	The velocity decay coefficient.
FIXED	Keyword that will tie the contacts together. (This keyword can be replaced by the value -1.)
toler	The tolerance for determining fixed contact. [Default = 0.02]

Description

Use this command to define contact between two surfaces. This command now has three distinct uses: a paired side set contact; a global contact; and fixed paired contacts. The paired side set contact is unchanged in its algorithm logic and usage. The global contact uses the new global contact detection algorithm and can be used to model a self-contacting surface (for example, see Contact Material).

For paired side set contact cases, the contact condition is enforced between the two surfaces defined by the respective *side sets*. The kinematic partition factor (β) is a relative weighting of the master-slave relationship of the two surfaces. A value of zero (0.0) implies that the first surface (defined by *side 1 id*) acts only as a master, and the second surface acts only as a slave. A value of one (1.0) reverses these roles. The default value (0.5) gives a symmetric treatment of the contact. If one surface is much more massive than the other, β should be adjusted so that the more massive surface is treated as the master. By massive it is meant that the surface either has a higher material density and/or a coarser mesh refinement.

A global contact condition is enforced between a surface contacting itself and other surfaces defined using the single side set Contact Surface keyword and those surfaces defined using the Contact Material keyword.

Example 1

```
$ format 1 (paired contact)
Contact Surface 1 2
```

Defines a contact between the surface defined by *side set id* 1 and *side set id* 2 with zero friction and equal partitioning (symmetric treatment of contact).

Example 2

```
$ format 1 (paired contact)
Contact Surface 1 2 .1
```

Defines a contact between the surface defined by *side set id* 1 and *side set id* 2 with a static coefficient of friction of 0.1 and equal partitioning (symmetric treatment of contact).

Example 3

```
$ format 1 (paired contact)
Contact Surface 1 2 .1 1
```

Defines a contact between the surface defined by *side set id* 1 and *side set id* 2 with a static coefficient of friction of 0.1. In this case the master surface will be *side set id* 2 and the slave surface will be *side set id* 1.

Example 4

```
$ format 2 (Fixed contact)
Contact Surface 1 2 Fixed 0.5 .001
```

Ties together the surfaces defined by *side set id* 1 and *side set id* 2. The tolerance value allows the small initial gaps or penetrations of nonmatching, curved surfaces to be ignored.

Example 5

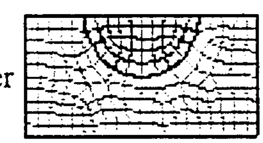
```
$ format 3 (global contact)
Contact Surface 1
Contact Surface 2
```

Flags the surfaces defined by *side set id* 1 and *side set id* 2 for global contact consideration. This would result in contacts being detected between surface 1 and itself, surface 1 and surface 2, surface 2 and itself, and surfaces 1 and 2 with any materials defined by the Contact Material keyword.

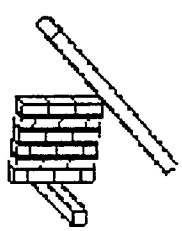
<u>**Problems**</u>

Cask Impacting Rail

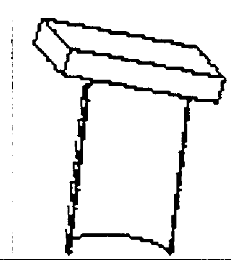
Contact Chatter



Brick Wall



Can Crush



Contact Material

Command Format

Contact Material *material id*

Parameters

material id — This value must match a material id on the GENESIS file. If no material id is specified, then all materials are included.

Description

Use this command to select a material to be evaluated for possible contact. All surfaces associated with *material id* are automatically determined and considered for self-contact and for contact with other surfaces defined by the single side set Contact Surface keyword and any additional surfaces defined as a result of repeated use of the Contact Material keyword. If the *material id* has an element Death option, the surface will be automatically redefined as elements die. Note that the automatic surface redefinition is only done for those surfaces defined by the Contact Material keyword and not those defined by the Contact Surface keyword.

Example 1

```
Contact Material
```

This simple command will determine the exterior surfaces of all materials in the problem definition and consider them for global contact.

Example 2

```
Contact Material 1
Contact Material 2
```

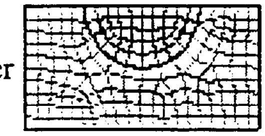
Here only materials with id 1 and id 2 will be added to the contact search.

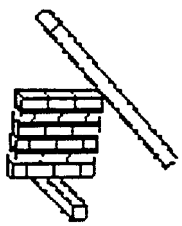
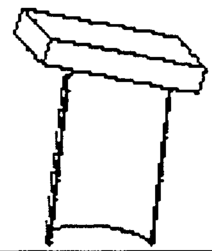
Example 3

```
Contact Material 1
Contact Surface 100
```

Material with id 1 and any element faces defined by *side set id* 100 will be added to the contact surface search. Note that if the FE surfaces in the *side set id* 100 overlap those on material 1, (i.e., some FE surfaces are defined twice), then the surfaces in *side set id* 100 take precedence, therefore, disallowing death of the FE surface.

Problems

Cask Impacting Rail  Contact Chatter 

Brick Wall  Can Crush 

Contact Exclude

Command Format

Contact Exclude *ss_id*

Parameters

ss_id	Side set id that will be removed from global contact.

Description

This command removes all faces specified by the side set **ss_id** from the global contact algorithm.

The global contact algorithm has always allowed the user to specify contact surfaces by an entire material block (Contact Material) and/or side sets (Contact Surface). Instances have arisen where the user would like to be able to specify all of a material with the exception of some small part (e.g., part of the material on a symmetry plane). The Contact Exclude command allows the user to specify a side set on which all faces will NOT be included in the contact algorithm. The Contact Exclude command has the highest precedence of all contact commands (i.e., it will override contact material for faces in that side set).

Contact Recompute Concavity

Command Format

Contact Recompute Concavity *n*

Parameters

n	Number of time steps between recomputing surface concavity. [Default is never]

Description

The global contact algorithm (Contact Material, Contact Surface) makes contact enforcement decisions based on the geometry of the contact surface. One of the many decisions is based on whether a surface "edge" is concave or convex. In the past, this was always computed at the start of the problem (or upon restart) and never updated. This command allows the user to specify how often this should be recomputed (**n** is the number of time steps between recomputing). The process of recomputing this quantity requires a communication step in parallel. As a result, the default behavior is to never update the concavity/convexity. As this feature is used and its effect on the solution is determined, the default behavior may be changed.

Contact Data

Command Format 1

Contact Data *id 1, id 2, Keyword=value, Keyword=value, ...*
end

Command Format 2

Contact Data *Defaults, Keyword=value, Keyword=value, ...*
end

Parameters

id 1	This value must be either "Surface surf id" or "Material mat id". "surf id" must match one of surfaces listed in the Contact Surface (format 3) command. "mat id" must match one of the materials listed in the Contact Material command.

id 2	This value must be either "Surface surf id" or "Material mat id".
Keyword=value	List of keywords followed by a value. The keywords can be in any order. The allowable keywords are listed below.
Defaults	Keyword that designates that the user wants to override the default values for selected contact data parameters.

Keywords

Friction Static	The static coefficient of friction. [Default = 0.0]
Kinematic Partition	The kinematic partition factor. [Default = 0.5]
Friction Dynamic	The high velocity coefficient of friction. [Default = 0.0]
Friction Decay	The velocity decay coefficient. [Default = 0.0]
Pushback Factor	The factor applied to the contact penetration. [Default = 1.0]
Capture Tolerance	A distance that is used in the contact search to capture slave nodes (i.e., considers the master surface-slave node pair for potential interaction). Note, if a slave node is farther than the capture tolerance from the master surface, it will not be considered for contact. [Default = 1.e-8]
Force Tolerance	The maximum force (always positive) that the contact interface will support without separating. [Default = 0.0]

Description

Use this command to define parameter values associated with the contact between surfaces or materials. All surfaces associated with the surface id/material id pair use the data specified by the Keyword list. If a part of a surface is multiply defined by a material id and a surface id, the contact data specification using the surface id will override the material id specification.

See Contact Data (old format) for compatibility with old input decks. This new keyword format is more readable and easier to expand.

Example 1

```
Contact Data MAT 1 MAT 2
   Friction Static = .1
End
```

Defines a static coefficient of friction of 0.1 between material *id 1* and material *id 2*.

Example 2

```
Contact Data Material 1 Material 2
   Friction Dynamic = .05
   Friction Static = .1
   Friction Decay = .01
   Kinematic Partition = 1.
End
```

Defines dynamic and static coefficients of friction between material *id 1* and material *id 2* with material *id 2* treated as the master surface.

Example 3

```
Contact Data Surface 1 Material 2
   Capture Tolerance = 0.001
   Friction Static = -1.0
   Kinematic Partition = 1.
End
```

Defines a fixed contact between surface *id 1* and material *id 2* with material *id 2* treated as the master surface. Note that the search for fixed contact is done once in the beginning of the analysis and will use a search tolerance of 0.001 as indicated in the Capture Tolerance keyword.

Contact Data (old format)

Command Format

Contact Data *id 1, id 2,* μ_0, β, μ_1, γ

Parameters

id 1 — This value must be either "Surface surf id" or "Material mat id". "surf id" must match one of surfaces listed in the Contact Surface (format 3) command. "mat id" must match one of the materials listed in the Contact Material command.

id 2 — This value must be either "Surface surf id" or "Material mat id".

μ_0	The static coefficient of friction. [Default = 0.0]
β	The kinematic partition factor. [Default = 0.5, JAS_Default = 0.0]
μ_1	The high velocity coefficient of friction. [Default = 0.0]
γ	The velocity decay coefficient.

Description

Use this command to define parameter values associated with the contact between surfaces or materials. All surfaces associated with the surface id/material id pair use the contact friction conditions and kinematic partitioning factor specified. If a part of a surface is multiply defined by a material id and a surface id, the contact data specification using the surface id will override the material id specification. See also Contact Data (new format).

Examples

```
Contact Data Material 1 Material 2 .1
```

Defines a static coefficient of friction of 0.1 between material *id 1* and material *id 2*.

Contact Window

Command Format

Contact Window *xmin, xmax, ymin, ymax, zmin, zmax*

Parameters

xmin	This value defines the minimum x-coordinate of the contact window. Slave nodes with x-coordinate below *xmin* will not be considered for contact.
xmax	This value defines the maximum x-coordinate of the contact window. Slave nodes with x-coordinate above *xmax* will not be considered for contact.
ymin	This value defines the minimum y-coordinate of the contact window. Slave nodes with y-coordinate below *ymin* will not be considered for contact.
ymax	This value defines the maximum y-coordinate of the contact window. Slave nodes with y-coordinate above *ymax* will not be considered for contact.

zmin — This value defines the minimum z-coordinate of the contact window. Slave nodes with z-coordinate below *zmin* will not be considered for contact.

zmax — This value defines the maximum z-coordinate of the contact window. Slave nodes with z-coordinate above *zmax* will not be considered for contact.

Description

Use this command to define a region in space inside of which contact will be determined.

Examples

```
Contact Window 0.0 1.0 0.0 1.0 0.0 1.0
```

Defines a unit cube in which contact will be determined.

Contact Level

Command Format

Contact Level *ivalue*

Parameters

ivalue — An integer value defining the level of contact treatment. [Default = 1]

Description

Use this command to specify the level of contact treatment. Level 1 is the simplest and most efficient, while level 2 is more robust but also more computationally expensive. Level 1 is a global contact search between the master surfaces and slave nodes defined in the Contact Surface and Contact Material commands and a single pass through the contact enforcement algorithm. Level 2 is a global contact search between the master surfaces and slave nodes defined in the Contact Surface and Contact Material commands and multiple passes through the contact enforcement algorithm. Level 2 will automatically determine the kinematic partitioning factor based on physical properties of the contacting surfaces. This work is in progress.

Example

```
Contact Level 2
```

Defines a level 2 contact treatment.

Material

Command Format

Material *material id, model,* ρ

Parameters

material id — This value must match an element block on the GENESIS file.

model — A valid material model name.

ρ — The material density.

Description

Use this command to specify the material model to be used for the specified material block. The material models currently supported in PRONTO3D are listed below. Only Elastic and Elastic Plastic models are currently available for shell elements.

Material Models Supported by PRONTO3D

Rigid	Soil N Foams	Thermoelastic
Elastic	EP Temp Depend	Wire Mesh
Elastic Plastic	EP Hydrodynamic	Sandia Damage
Viscoplastic	EP Power Hardening	Orthotropic Crush
Damage	Johnson Cook	PLH Strength
Hydro	Hyperelastic	New Foam
Low Density Foam	Thorne Damage	

Appropriate material data for the given material model must be entered immediately following the Material command line. The data is entered in a keyword/value fashion, with a material cue keyword followed by its assigned value. Each material type requires its own set of material cues. The material cues can be entered in any order and on any number of input lines. An END statement is required to terminate the material data. The required material cues are also listed in Table 3. Consult the PRONTO3D manual [Taylor, L.M. and Flanagan, D.P., 1989] for definitions of the material parameters.

Examples

Examples of how the user might input the material data for the Elastic Plastic model are given below. They illustrate several different styles. All four examples yield identical results as far as PRONTO3D is concerned.

Example 1

```
Material 1, Elastic Plastic, 2.7E-4
   Hardening Modulus = 30.E4
   Youngs Modulus = 30.E6
   Beta = .5
   Poissons Ratio = .3
   Yield Stress = 30.E3
End
```

Example 2

```
Material 1, Elastic Plastic, 2.7E-4
   Youngs Modulus=30.E6, Poissons Ratio=.3, Beta=.5
   Yield Stress=30.E3, Hardening Modulus=30.E4
End
```

Example 3

```
Material 1,Elastic Plastic,2.7E-4
   Youngs Modulus = 30.E6,
   Poissons Ratio = .3, Beta = .5
   Yield Stress = 30.E3, Hardening -
Modulus = 30.E4, End
```

Example 4

```
$ {ECHO (OFF)} {Units("in-lbf-s")}
$ {ECHO (ON)}

Material {matsteel=1} ,Elastic Plastic, -{2.8854~g/cm^3}
   Youngs Modulus {2.068427188e+5~MPa}
   Poissons Ratio = .3
   Beta = .5
   Yield Stress = {30~ksi}
   Hardening Modulus = {30.E4~psi}
End
```

Equation of State

Command Format

Equation of State *material id, eos*

Parameters

material id This value must match an element block on the GENESIS file.

eos A valid equation of state model name.

Description

Use this command to specify the equation of state to be used for a specified material block. The EOS models currently supported in PRONTO3D are MG US-UP, MG POWER SERIES, JWL, IDEAL GAS.

Appropriate material data for the given equation-of-state model must be entered immediately following the Equation of State command line. The data is entered in a keyword/value fashion, with a material cue keyword followed by its assigned value. Each model requires its own set of material cues. The material cues can be entered in any order and on any number of input lines. An END statement is required to terminate the material data.

The required material cues for the currently supported equation-of-state models are listed in the table below. Consult Chapter 5 of the PRONTO3D manual [Taylor, L.M. and Flanagan, D.P., 1989] for definitions of these parameters.

Equation-of-State Material Cues

Equation-of-State Models	Parameters
MG US-UP	C0 S Gamma
MG POWER SERIES	K0 K1 K2 Gamma
JWL	Gamma CD A B Omega R1 R2 Energy
IDEAL GAS	Gamma Sound Speed

Example

An example material for the MG US-UP equation of state is given below. The Material command using the HYDRO material name is also shown. Note that the "material id" on the Material command matches the "material id" on the Equation of State command.

```
Material 8, Hydro, 2.7E-3
   Pressure Cutoff = -1.E9
$(note: pressure negative in tension!)
End

Equation of State 8, MG US-UP
   C0 = 5380, S = 1.337, GAMMA = 2
End
```

Detonation Point

Command Format

Detonation Point *material id*, c_x, c_y, c_z, t_0

Parameters

material id	The material number of the high explosive to be detonated. This value must match an element block on the GENESIS file.
c_x, c_y, c_z	The detonation point coordinates.
t_0	The detonation time.

Description

Use this command to define the location and time of detonation associated with a specified material block. The Jones-Wilkins-Lee or JWL Equation of State provides the pressure generated by the release of chemical energy in an explosive. In PRONTO3D it is implemented in a form that is usually referred to as a programmed burn. A programmed burn means that the reaction and initiation of the explosive is not determined by the shock in the material. Rather the initiation time is determined by a Huygens construction using the detonation wave speed and the distance of the material point from the detonation point(s).

The programmed burn requires the initial calculation of the arrival of the detonation wave at a material point. If there is only one detonation point, denoted by x_d, and if the location of the material point is denoted by x_n, then the detonation time is determined by

$$t_d = \frac{\|x_d - x_n\|}{c_d} \quad (9)$$

where c_d is the detonation wave speed (a material property supplied by the user for the JWL Equation of State), and the symbol $\|\ldots\|$ indicates the Euclidean norm of a vector. If there are multiple detonation points, then detonation time will be determined from the minimum detonation time.

Example 1

```
Detonation Point 10 1. 0. 0. 0.0003
```

A single point detonation located at (1., 0., 0.) and starting at time $t_0 = 0.0003$.

Example 2

```
$ {ECHO OFF}
{xstart = 0} {xend = 10} {ndiv = 10}
{delta=(xend-xstart)/ndiv}
{x = xstart-delta}
{ECHO ON}

{Loop ndiv}
Detonation Point 10 {x=x+delta} 0. 0. 0.0003
{End Loop}
```

The example above shows how to create a multipoint detonation wave using APREPRO. Ten equally spaced detonation points along the line $0 < x < 10$, $y = 0$, $z = 0$ will start at time $t_0 = 0.0003$.

Burn Constant

Command Format

Burn Constant *bs*

Parameters

bs	The high explosive burn constant. [Default = 2.5]

Description

Use this command to specify the value of the burn constant. The Jones-Wilkins-Lee or JWL Equation of State provides the pressure generated by the release of chemical energy in an explosive. In PRONTO3D it is implemented in a form that is usually referred to as a programmed burn. A programmed burn means that the reaction and initiation of the explosive is not determined by the shock in the material, rather the initiation time is determined by a Huygens construction using the detonation wave speed and the distance of the material point from the detonation point(s).

To spread the burn wave over several elements, a burn fraction F is computed as

$$F = \min[1, \frac{(t - t_d)c_d}{B_s l}] \tag{10}$$

where B_s is a constant that controls the width of the burn wave, l is the characteristic length of the element, c_d is the detonation wave speed, t is the current problem time and t_d is the detonation time (time of arrival of the detonation wave at the element).

Example

```
Burn Constant 3.0
```

Delete Material

Command Format

Delete Material *material id, deletion time*

Parameters

material id	This value must match an element block on the GENESIS file.
deletion time	The time when all elements in this material block are deactivated.

Description

Use this command to delete the specified material at the specified time.

The element variable STATUS will be placed on the EXODUS data file to indicate whether the element is active or inactive. A STATUS value equal to 1.0 indicates an inactive element.

If the material is one of the materials in the Contact Material definition, then the contact surface will be redefined at the time of deletion to reflect the removed material.

Example

```
Delete Material 10 0.001
```

Death

Command Format

Death *material id, variable name, mode, level, eval, steps*

Parameters

material id	This value must match an element block on the GENESIS file.
variable name	The name of the critical variable. The critical variable may be any one of the element or state variables listed in Table 1.
mode	The criticality mode. This value may be MIN (minimum value), MAX (maximum value), or ABS (absolute value).
level	The critical value.
eval	The energy release rate. [Default = 0.0]
steps	The number of time steps over which the element dies. [Default = 5]

Description

Use this command to control the death of elements in the specified material block. The adaptive element deletion capability requires an experienced user who understands how the selected material behaves. The capability built into PRONTO3D is quite general and allows elements to be deleted depending upon the level of energy, von Mises stress, pressure, maximum principal stress, or any of the internal state variables for the material model. Any of the element variables listed in the Print Info command can be used in the Death command.

The criticality mode determines how failure occurs. The MIN *mode* signifies failure when the value of the critical variable falls below the critical level. Failure occurs in the MAX *mode* when the value of the critical variable exceeds the critical level. ABS *mode* is similar to MAX *mode*, except that the absolute value of the critical variable is used.

The user should be aware that it is possible to define nonsensical data by using a *mode* specification that is inappropriate for the critical variable. An example of this would be using the MIN specification with the VonMises variable and a negative level.

Care must be taken to avoid deleting elements that have side boundary conditions (Pressure or paired Contact Surface) applied to them. The Contact Material command must be used if the eroded surface is to be updated and included in the contact algorithm.

Examples

Syntax

```
Death 3, DAMAGE, MAX, 0.8
```

This command would delete elements within the material block with *material id* 3 in which the damage exceeds a value of 0.8. Note that PRONTO3D would insist that this material block use the Damage material model. The default value of five (5) time steps would be used to kill the element, with no energy absorbed during the element's death.

Problem

Shell Tearing 

Gravity

Command Format

Gravity *function id*, g_x, g_y, g_z, *scale*

Parameters

function id	The Function used to scale the gravity load as a function of time.
g_x, g_y, g_z	The relative strength of gravity in the x, y, and z directions.
scale	Scale factor for the function. [Default = 1]

Description

Use this command to apply a gravity load as a funtion of time. Applying an instantaneous gravity load can cause unwanted dynamic behavior. If the load is ramped over time and the ramp time is less than the fundamental period of the problem, the dynamic effects can be kept to a minimum.

An instantaneous gravity load can be applied if an initial stress that balances the gravity load is also applied.

Example

```
$ ramp gravity load
Function 10
   0. 0.
   0.001 1.
   1. 1.
end
Gravity 10 0. 9.8 0.
```

Ramps the gravity load from zero to 9.8 in 0.001 time units.

See the Initial Value command description for an example that shows how to set a hydrostatic pressure.

SPH

Command Format

SPH

Parameters

None

Description

Use this command to turn sphere elements (as defined on the GENESIS data file) into smooth particle hydrodynamics elements and to activate the SPH algorithm. Any of the commands that start with SPH will activate the SPH algorithm so that this command need not be given if any of the other SPH commands are given. See the following for details: [Swegle, J.W., Attaway, S.W., Heinstein, M.W., Mello, F.J. and Hicks, D.L., 1993]; [Guenther, C., Hicks, D.L. and Swegle, J.W., 1994] and [Wen, Y., Hicks, D. L. and Swegle, J. W., 1994].

Example

```
SPH
```

SPH Viscosity

Command Format

SPH Viscosity *type, visc1, visc2, id 1, id 2*

Parameters

type	The type of viscosity to be applied to SPH elements (Literature or VNR).
visc1	Literature: alpha, VNR: b1 - quadratic VNR term. [Default: alpha = 1., b1 = 0]
visc2	Literature: beta, VNR: b2 - linear VNR term. [Default: beta = 2., b2 = 0]
id 1	Must match a material id on the GENESIS database. (If left blank, then the *visc1* and *visc2* values will apply to all SPH materials.)
id 2	Must match a material id on the GENESIS database.

Description

Use this command to specify the type and values of the SPH viscosity. Two types of SPH Viscosity are available: Literature or VNR. If Literature is specified, then the so-called Monaghan viscosity is applied. If VNR is specified, then the code uses a von Neumann-Richtmeyer viscosity. Both Literature and VNR viscosities can be used at the same time, with the total viscosity being the sum of the two different viscosity algorithms. See the following for details: [Swegle, J.W., Attaway, S.W., Heinstein, M.W., Mello, F.J. and Hicks, D.L., 1993]; [Guenther, C., Hicks, D.L. and Swegle, J.W., 1994] and [Wen, Y., Hicks, D. L. and Swegle, J. W., 1994].

Example 1

```
SPH Viscosity LITERATURE .5 1.
```

The above command will assign Monaghan viscosity coefficients alpha = 0.5 and beta = 1.0 as the default for all SPH elements.

Example 2

```
SPH Viscosity LITERATURE .5 1.
SPH Viscosity LITERATURE .5 .5 1 2
```

The above will assign Monaghan viscosity coefficients alpha = 0.5 and beta = 1.0 as the default for all SPH elements. Between materials *id 1* and *id 2*, the defaults will be overridden, and values alpha = 0.5 and beta = 0.5 will be used instead.

Example 3

```
SPH Viscosity LITERATURE .5 1.
SPH Viscosity VNR .5 .5 1 1
```

The above will assign Monaghan viscosity coefficients alpha = 0.5 and beta = 1.0 as the default for all SPH elements except for *material id* 1. For *material id* 1, von Neumann-Richtmeyer viscosity, with quadratic term b1 = 0.5 and linear term b2 = 0.5, will be used.

SPH Viscosity Timestep

Command Format

SPH Viscosity Timestep

Parameters

None

Description

Use this command to include the Monaghan viscosity coefficients alpha and beta (see SPH Viscosity) in the viscosity parameters used in the time step calculation. Default is not to include them, so the quadratic term is b1**2, and the linear term is b2. If this command is given, then the quadratic term is b1**2+beta, and the linear term is b2+alpha/3. See the following for details: [Swegle, J.W., Attaway, S.W., Heinstein, M.W., Mello, F.J. and Hicks, D.L., 1993]; [Guenther, C., Hicks, D.L. and Swegle, J.W., 1994] and [Wen, Y., Hicks, D. L. and Swegle, J. W., 1994].

Example

```
SPH Viscosity Timestep
```

SPH Velocity Smoothing

Command Format

SPH Velocity Smoothing *b4*

Parameters

b4 — Conservative smoothing parameter (must be between 0 and 1 with 0 = no smoothing, 1 = maximum smoothing).
[Default = 0, no smoothing]

Description

Use this command to turn on conservative smoothing. See the following for details: [Guenther, C., Hicks, D.L. and Swegle, J.W., 1994] and [Wen, Y., Hicks, D. L. and Swegle, J. W., 1994].

Example

```
SPH Velocity Smoothing .5
```

SPH Interface Smoothing

Command Format

SPH Interface Smoothing *id1, id2*

Parameters

id1, id2 — The material ids for which conservative smoothing will be applied across the interface between the two materials. *id1* and *id2* must match material ids on the GENESIS database.
[Default is no smoothing across interfaces between different materials].

Description

Use this command as a flag for smoothing between particles of materials *id1* and *id2* when SPH Velocity Smoothing has been used, and b4 is not equal to zero.

Example

```
SPH Interface Smoothing 1 2
```

SPH Decouple Strains

Command Format

SPH Decouple Strains *id1, id2*

Parameters

id1, id2 The material ids for which strains will be decoupled across interfaces. *id1* and *id2* must match material ids on the GENESIS database. [Default is to couple the strains across material interfaces].

Description

Use this command to decouple the strains across material interfaces. When this option is used, material penetration is prevented by the viscosity generated from SPH Viscosity (Literature option).

Since SPH calculates strains based on the average strain in the neighborhood of a material, it is sometimes better to treat dissimilar materials, such as air and steel, separately. See [Swegle, J.W., Attaway, S.W., Heinstein, M.W., Mello, F.J. and Hicks, D.L., 1993] for details.

Example

```
SPH Decouple Strains 1 2
```

SPH Variable Smoothing

Command Format

SPH Variable Smoothing *type*

Parameters

type Designates the type of SPH Variable Smoothing to be used. [Default = 0]

Description

Use this command to define the SPH variable smoothing type. The smoothing length of a SPH element, h, can change with time as a function of the element density. This command allows the user to select how the new values of h are computed.

type = 0 signifies a constant h.

type = 1 signifies a variable h estimating h at $t_{n+3/2}$ for strains.

type = 2 signifies a variable h using h at t_n for strains.

See [Swegle, J.W., Attaway, S.W., Heinstein, M.W., Mello, F.J. and Hicks, D.L., 1993] for details.

Example

```
SPH Variable Smoothing 2
```

SPH Kernel Density

Command Format

SPH Kernel Density

Parameters

None	[Default is to use the continuity equation to compute the change in density]

Description

Use this command to calculate densities from the kernel sum rather than from the solution of the continuity equation. See [Swegle, J.W., Attaway, S.W., Heinstein, M.W., Mello, F.J. and Hicks, D.L., 1993] for details. Also see SPH Density Normalization.

Example

```
SPH Kernel Density
```

In general, for gas and explosive by-products, the kernel sum density gives better results. For solids, the continuity equation tends to give better results.

SPH Scale Factor

Command Format

SPH Scale Factor *h*

Parameters

h	The scale factor for the smoothing length. [Default = 1.0]

Description

Use this command to set the SPH smoothing length scale factor. Each SPH element has a smoothing length that determines its radius of influence. The initial size is read from the GENESIS data file as an attribute of the element. The scale factor will adjust the initial smoothing length (size) of the element. If SPH Variable Smoothing is used, then the smoothing length will also be a function of density, allowing the size to adapt.

Example

```
SPH Scale Factor 1.5
```

SPH Symmetry Plane

Command Format

SPH Symmetry Plane *isym, xsym*

Parameters

isym	A single symmetry plane normal to a coordinate axis can be specified. The value of *isym* chooses the axis as follows: 1 = x axis 2 = y axis 3 = z axis
xsym	Position (coordinate) of the symmetry plane on the specified axis.

Description

Use this command to define the SPH symmetry plane. A single symmetry plane is currently allowed in PRONTO3D. SPH elements will be mirrored across the symmetry plane.

Example

```
SPH Symmetry Plane 1 0.
```

SPH Density Normalization

Command Format

SPH Density Normalization

Parameters

None

Description

Use this command to normalize kernel densities at each time by a factor that sets the kernel density to ambient density at `t` = 0. Also see SPH Kernel Density.

Example

```
SPH Kernel Density
SPH Density Normalization
```

Print Info

Command Format

Print Info

Parameters

None

Description

Use this command to print information for all nodal and element variables used in the current problem. All component names, their sizes, and memory addresses are summarized. Any of the names listed in the Print Info command can be output to the EXODUS data file using the Plot Nodal, Plot Element, Plot State, or the Plot History commands.

Example

```
Print Info
```

Initial Value

Command Format 1

Initial Value *component, value, Material matid*

Command Format 2

Initial Value *component, value, Function, dir, function id*

Parameters

component	The full name of the variable component to be initialized.
value	The initial value of component. (May be overwritten by Read Restart.)
Material matid	If the keyword *Material* is present, then the initial value will apply only to *matid*. If the keyword *Material* is omitted, then all materials, nodes, and shell nodes will be searched to see if they have a variable with the name *component*.
Function	If the keyword *Function* is present, then the *component* will be set as a function of location.
dir	Defines the coordinate which is to be used as input to the function. Possible values are X, Y, Z and R.
function id	This value must match a *function id* defined using keyword Function.

Description

Use this command to set initial values for the specified variables. It is intended to help constitutive modelers who would like to preset initial values of state variables. The command will allow a user to set the initial value of any element or nodal variable. State variables are treated as element variables. For a complete list of variable names available, use the Print Info command.

This command may get you into big trouble. It takes an experienced developer to know when the Initial Value command can be used. For example, if you try to set the initial velocity with this command, then you will not get the correct answers. The initial velocity command actually is converted into an initial acceleration at the start of the first half time step. To minimize the chance for errors, users should set values using other PRONTO3D Commands as listed in this manual.

Example 1

```
Initial Value DAMAGE, 0.1
Initial Value DAMAGE, 0.9, MATERIAL, 3
```

In this example the initial state variable Damage will be set to 0.1 for all materials, with material 3 having an initial value of 0.9. Here, the order the command input is important. If the order is reversed, then all materials would have a damage value of 0.1.

Example 2

```
$ Example: initial stress and initial gravity
${Units('SI')}
Initial Value density function z 1 mat 100
Initial Value usigzz function z 2 mat 100

$ { gravity = 980.612e-12*cm/usec**2}
$ { rho0 = 1.0*gpcc}
$ { speed0 = .165*cm/usec}
$ { z0depth = 6132.88*cm}

$ Density Function
Function 1 polynomial
{a0 = rho0*(1.+gravity*z0depth/speed0**2)}
{a1 = -rho0*gravity/speed0**2}
end
$ gravitational stress - negative pressure
Function 2 polynomial
{a0 = -rho0*gravity*z0depth}
{a1 = rho0*gravity}
end

Gravity 10 0 0 {-gravity}
$ Gravity function
Function 10
   0        1
   {1*year} 1
end
```

The example above shows how to use the Initial Value command with the Gravity and Function commands to initialize the density and z component of the unrotated stress for material 100 to a hydrostatic pressure that is linear with depth. APREPRO is used to allow the initial density, wave speed, and depth to be input as parameters.

Cavity Expansion

Comand Format

Cavity Expansion *side set id, AXIS=direction, BOUNDS=b1, b2, COEF=c1, c2, c3*

Parameters

side set id	This value must match a side set id on the GENESIS file.
AXIS	A keyword that specifies that the next value to be the direction of the cavity axis.

direction	X, Y, or Z [default = Z]
BOUNDS	A keyword that specifies that the following values to be the bounding coordinates.
b1, b2	The bounding coordinates along the direction specified in AXIS.
COEF	A keyword that specifies that the following values are the nodal pressure coefficients.
c1, c2, c3	The constant nodal pressure coefficients.

In certain penetration events the primary mode of deformation of the target can be approximated by known analytical expressions. The spherical Cavity Expansion forcing function is implemented as a normal traction (or pressure) boundary condition that acts on a prescribed surface.

The command line shown above can be repeated, as needed, in order to represent layers with different parameters.

Analytical methods for penetration mechanics began with the work of [Bishop, R.F., Hill, R., and Mott, N.F., 1945]. They developed equations for the quasi-static expansions of cylindrical and spherical cavities and used these equations to estimate forces on conical nose punches pushed slowly into metal targets. [Goodier, J.N. 1965] developed a model to predict the penetration depth of rigid spheres launched into metal targets. His penetration model included target inertial effects, so he approximated the target response by results from the dynamic, spherically symmetric, cavity-expansion equations for an incompressible target material derived by [Hill, R., 1948] and discussed by [Hill, R., 1950] and [Hopkins, H.G., 1960]. The method used here follows the more recent work of Forrestal ([Forrestal, M.J., Okajima, K., and Luk, V.K., 1988]; [Forrestal, M.J., Brar, N.S., and Luk, V.K., 1991]; [Forrestal, M.J., Tzou, D.Y., Askari, E., and Longcope, D.B., 1995]; [Forrestal, M.J., and Tzou, D.Y., 1996]; [Warren, T.L. and Forrestal, M.J., 1997]; and [Warren, T.L. and Tabbara, M.R., 1997]).

The radial stress at the cavity surface obtained from spherical cavity-expansion models can be accurately represented by a function of the form

$$\frac{\sigma_r(a)}{Y} = A + B\left(\sqrt{\frac{\rho_0}{Y}}\vartheta\right) + C\left(\sqrt{\frac{\rho_0}{Y}}\vartheta\right)^2 \qquad (11)$$

where ϑ is the target particle velocity at the cavity-target interface; a is the cavity radius; Y is the quasi-static yield strength of the target material; ρ_0 is the density of the undeformed target material; and A, B, and C are dimensionless fitting coefficients. The expression given in Equation (11) is also consistent with the semiempirical model developed by [Forrestal, M.J., Altman, B.S., Cargile, J.D., and Hanchak, S.J., 1994] for penetration into concrete targets.

Four nodal pressures are calculated in PRONTO3D for each element side (i.e., a side of a hexagonal continuum element or mid-surface of a structural shell element) included in the side set as shown in Figure 5. These nodal pressures are obtained from

$$p_I = c_1 + c_2(\underset{\sim}{V}_I \bullet \underset{\sim}{n}) + c_3(\underset{\sim}{V}_I \bullet \underset{\sim}{n})^2 \qquad (I = 1, 4) \tag{12}$$

where the dot represents a scalar product, $\underset{\sim}{V}_I$ is the nodal velocity vector, $\underset{\sim}{n}$ is the outward unit vector normal to the diagonals of the side, and the constant nodal pressure coefficients are related to the dimensionless fitting coefficients in Equation (11) as

$$c_1 = AY \tag{13}$$

$$c_2 = B(\rho_0 Y)^{\frac{1}{2}} \tag{14}$$

and

$$c_3 = C\rho_0 \tag{15}$$

The values of p_I are updated during each time increment using the current values of $\underset{\sim}{V}_I$ and $\underset{\sim}{n}$. If the scalar product $(\underset{\sim}{V}_I \bullet \underset{\sim}{n})$ at a node is zero, negative, or if the node lies outside the bounds set by *b1* and *b2*, then the pressure is set to zero for that node.

A set of consistent global forces arising from these pressures over an element side are calculated as discussed by [Taylor, L.M. and Flanagan, D.P., 1987]. These forces are accumulated as each element side in the side set is considered.

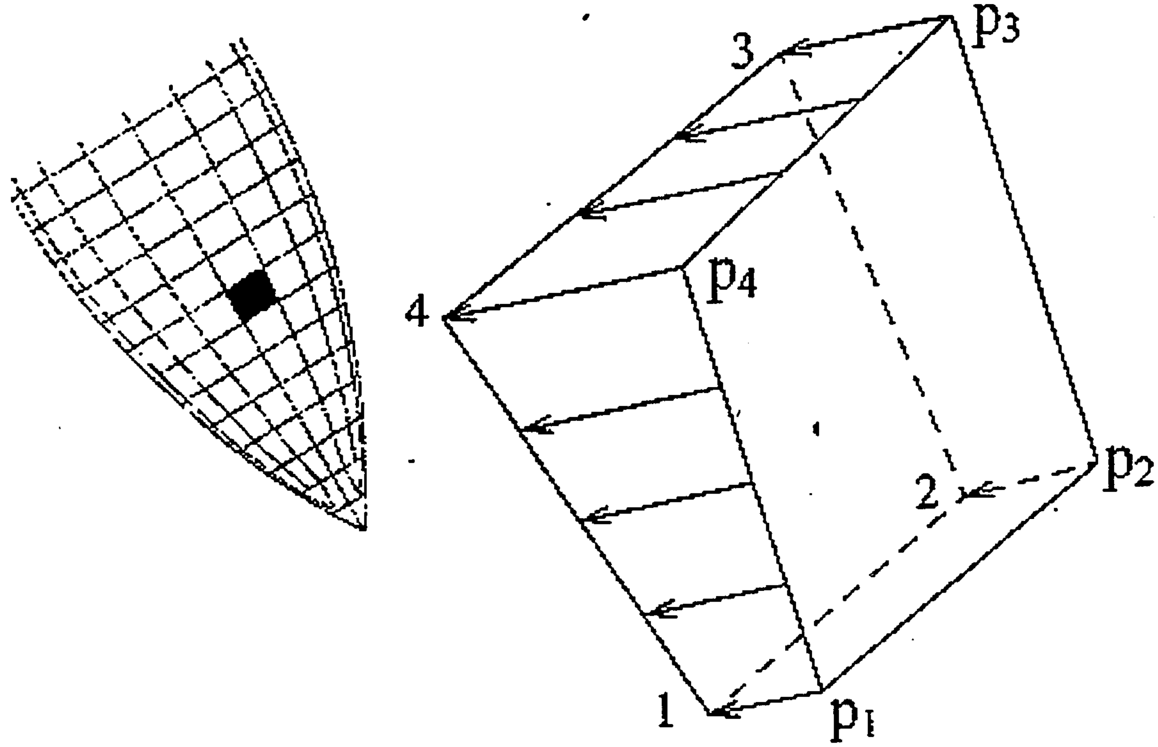


Figure 5 **Definition of a pressure boundary condition that acts on an element side.**

[Forrestal, M.J., Okajima, K., and Luk, V.K., 1988] recognized that the resistance produced by an aluminum target could be approximated by a dynamic cavity-expansion analysis. They developed closed-form expressions for the depth of penetration of rigid projectiles with different nose shapes and demonstrated good agreement with experimental results. The same concept is applied here, but in the context of a three-dimensional finite element code. This implementation is capable of handling a full three-dimensional penetration event that includes: oblique impact, nonzero angle of attack, nonlinear deformations of the projectile, response of components internal to the projectile, etc. The accuracy of this method depends on how well the forcing function approximates the actual situation; however, in many cases the spherical cavity expansion (which is derived on the basis of an unbounded medium) does provide a good approximation for events where the free surface effects are minimal. Thus, this implementation is most accurate for cases of deep penetration.

This method has previously been applied with some success using cavity expansion forcing functions with beam elements in the general purpose finite element code ABAQUS [Hibbitt, Karlsson and Sorensen, Inc., 1989] implicitly by [Longcope, D.B., 1991] and [Longcope, D.B., 1996], using empirical forcing functions with shell elements in ABAQUS implicitly by [Adley, M.D., and Moxley, R.E., 1996], with shell elements in ABAQUS explicitly by [Duffey, T.A., and Macek, R.W., 1997], and also with tetrahedron, brick, and shell elements in EPIC 97 ([Johnson, G.R., Stryk, R.A., Holmquist, T.J., and Beissel, S.R., 1997]).

Examples

Cavity Expansion: Concrete

Cavity Expansion: Aluminum

Energy Deposition

Command Format 1

Energy Deposition *External NAME=varname, RAMPT=deptime*

Command Format 2

Energy Deposition *Internal MATERIAL=matid, TFUNCTION=thid, SPACEFUNCTION=dir, sfunid, USERFUNCTIONS=funid1, funid2, TSCALE=scale*

Parameters

External	A keyword that specifies that the energy values should be read from a GENESIS file.
NAME	A keyword that specifies that the following argument is the variable name for energy per unit volume.
varname	The name that corresponds to the energy per unit volume on the EXODUS database. [Default = energy]
RAMPT	A keyword that specifies that the time variables other than time equal to zero on the GENESIS file are ignored, and the deposited energy is ramped linearly from zero to the value given by the **varname** variable on the GENESIS file over the time interval.
deptime	Defines the time interval over which the deposited energy is ramped (try 3 to 10 steps). [Default = 0]
Internal	A keyword that specifies that the energy values should be calculated as a function of time.
MATERIAL	A keyword that the following argument is the material block to which the energy is deposited.
matid	This value must match a material id on the GENESIS file.

TFUNCTION A keyword that specifies that the following value is the function id for the function to calculate the energy time history.

thid This value must match a function defined via Function.

SPACEFUNCTIONS A keyword that specifies that the following values are used to define the spatial distribution of the energy deposition.

dir X, Y, Z, or R. If *dir* = *X*, *Y*, or *Z*, then the energy deposition is scaled as a function of the appropriate coordinate using the function on the input file with function id **sfunid**. If *dir* = **R** then the energy is scaled as a function of distance from the origin using the function on the input file with function id **sfunid**.

sfunid This value must match a function defined via Function.

USERFUNCTIONS A keyword to specify that a user-provided subroutine should be used to scale the time history function for a material block in any desired manner.

funid1 This value must match a function defined via Function. It is passed to the user provide subroutine.

funid2 This value must match a function defined via Function. It is passed to the user provide subroutine.

TSCALE A keyword specifying that the following value is the scale factor applied to *SPACEFUNCTIONS*.

scale Scale factor applied to *SPACEFUNCTIONS*.

Description

NOTE: Both *funid1* and *funid2* must be included in the command for any material block for which the *USERFUNCTIONS* keyword is specified even if the user-provided subroutine does not require a function from the input file.

Energy deposition can be input either as a time history variable from an external source on the GENESIS file, or from a time history function. A variety of methods of scaling the deposited energy as a function of position are available.

If the *EXTERNAL* keyword is used, then the energy time history variable, *varname*, on the GENESIS file must be an energy time history given as energy per unit volume in the system of units used in the PRONTO3D calculation. If an initial value of energy is to be deposited at the start of a problem, then a GENESIS file can be read that has an energy value from an independent source at a single time. In this case the time variables other than time zero on the GENESIS file are ignored, and the deposited energy is ramped linearly from zero to the value given by the *varname* variable on the GENESIS file over a time interval given by *deptime* (usually 3 to 10 time steps).

If the *INTERNAL* keyword is used, then the energy time history given by the function on the input file with function id *thid* multiplied by the scale factor *sfunid* is used to deposit energy in the material block corresponding to *matid*. Clearly no material id should appear on more than one energy deposition command. However, no check for this input error is provided. If the value *X*, *Y*, or *Z* is present on an energy deposition command, then the energy deposition time history for material block *matid* is scaled as a function of the appropriate coordinate using the function on the input file with function id *spaceid*. If the value *R* is present, then the energy time history for material block *matid* is scaled as a function of distance from the origin using the function on the input file with function id *spaceid*. Provisions are made for a user-provided subroutine that can be used to scale the time history function for a material block in any manner desired. This option is activated by specifying the keyword *USERFUNCTION*. If the *USERFUNCTION* keyword is specified for a material block, provisions are made to pass two function ids, *funid1* and *funid2*, to the user provided subroutine. Both *funid1* and *funid2* must be included in the command for any material block for which the *USERFUNCTION* keyword is specified even if the user-provided subroutine does not require a function from the input file.

Subcylcing

Command Format

Subcylcing

Subcylcing allows each block of hex elements to be integrated with its own individual time step. In the PRONTO3D architecture, this means skipping the residual calculation for the larger time step element blocks while the smaller time step blocks are subcycled forward in time. Subcylcing results in a CPU savings factor of dT/dt in the residual calculation of larger time step blocks, which allows small features to be resolved without sacrificing computational efficiency. However, the reduction in CPU time is problem dependent; there is no saving if all elements are equal size. Also the savings is only in the residual calculation; other calculations, such as contacts and I/O, can take significant CPU.

Appendix (Tables)

Table 1 Nodal and Element Variable Names

Table 1a Nodes

Variable Aliases	Variable Names	Component Names
Coordinates	COORD	COORDX COORDY COORDZ (3D)
Displacements	DISPL	DISPLX DISPLY DISPLZ (3D)
Velocity	VEL	VELX VELY VELZ (3D)
Predicted Velocity	PVEL	PVELX PVELY PVELZ (3D)
Force	FORCE	FORCEX FORCEY FORCEZ (3D)
Acceleration	ACCL	ACCLX ACCLY ACCLZ (3D)
Nodal Mass	XMASS	XMASS
Reactions	REACT	REACTX REACTY REACTZ (3D)
Current Position	CUR	CURX CURY CURZ (3D)
Temperature	TEMP	TEMP (2D)

Table 1b Shell Nodes (3D)

Variable Aliases	Variable Names	Component Names
Nodal Basis	BASND	BASNDXX BASNDXY BASNDXZ BASNDYX BASNDYY BASNDYZ BASNDZX BASNDZY BASNDZZ
Rotational Displacement	ROTDIS	ROTDISX ROTDISY ROTDISZ
Rotational Velocity	ROTVEL	ROTVELX ROTVELY ROTVELZ
Rotational Acceleration	ROTACC	ROTACCX ROTACCY ROTACCZ
Rotational Mass	ROTMASS	ROTMASSX ROTMASSY ROTMASSZ
Moment	MOMENT	MOMENTX MOMENTY MOMENTZ
Nodal Status	NSTATUS	NSTATUS

Table 1c Quad Elements (2D)

Variable Aliases	Variable Names	Component Names
Stress	SIG	SIGXX SIGYY SIGZZ SIGXY
Energy	ENERGY	ENERGY
Element Mass	ELMASS	ELMASS
Hourglass Resistance	HGR	HGX HGY
Strain	EPS	EPSXX EPSYY EPSZZ EPSXY
Stretch	STRECH	STRECHXX STRECHYY STRECHZZ STRECHXY
Rotation	ROTATE	COSTHETA SINTHETA
Density	RHO	DENSITY
Viscous Pressure or Bulkq	VISPR	BULKQ
Rate Deformation or RATEDFM	DOPT	DOPTXX DOPTYY DOPTZZ DOPTXY
Hourglass Energy	HGE	HGENGY
Pressure	PRESS	PRESSURE
Von Mises	VONMIS	VONMISES
Element Temperature	TEMN	TEMN
State Variables	SV	Components are the names provided in MATINT for each material.

Table 1c Quad Elements (2D)

Variable Aliases	Variable Names	Component Names
EOS State	EOSSV	Components are the names created in EOSINT for each equation of state.
Status	STATUS	STATUS

Table 1d Hex Elements (3D)

Variable Aliases	Names	Component Names
Stress	SIG	SIGXX SIGYY SIGZZ SIGXY SIGYZ SIGZX
Energy	ENERGY	ENERGY
Element Mass	ELMASS	ELMASS
Hourglass Resistance	HGR	HG1X HG1Y HG1Z HG2X HG2Y HG2Z HG3X HG3Y HG3Z HG4X HG4Y HG4Z
Strain	EPS	EPSXX EPSYY EPSZZ EPSXY EPSYZ EPSZX
Stretch	STRECH	STRECHXX STRECHYY STRECHZZ STRECHXY STRECHYZ STRECHZX

Table 1d **Hex Elements (3D)**

Variable Aliases	Names	Component Names
Rotation	ROTATE	R11 R21 R31 R12 R22 R32 R13 R23 R33
Density	RHO	DENSITY
Viscous Pressure or Bulkq	VISPR	BULKQ
Rate Deformation or RATEDFM	DOPT	DOPTXX DOPTYY DOPTZZ DOPTXY DOPTYZ DOPTZX
Hourglass Energy	HGE	HGENGY
Pressure	PRESS	PRESSURE
Von Mises	VONMIS	VONMISES
State Variables	SV	Components are the names provided in MATINT for each material.
EOS State	EOSSV	Components are the names created in EOSINT for each equation of state.
Status	STATUS	STATUS

Table 1e Rigid Hex Elements (3D)

Variable Aliases	Variable Names	Component Names
Element Mass	ELMASS	ELMASS
Status	STATUS	STATUS

Table 1f **Shells (3D)**

Aliases	Variable Names	Component Names
Stress	SIG	SIGXX1 SIGYY1 SIGZZ1 . . . SIGXY*ninteg* SIGYZ*ninteg* SIGZX*ninteg*
Energy	ENERGY	ENERGY
Element Mass	ELMASS	ELMASS
Thickness	THICK	THICK
Total Thickness	TOTALT	TOTALT
Offset	OFFSET	OFFSET
Element Bases	BASEL	BASELXX BASELYY BASELZZ BASELXY BASELYZ BASELZX
Hourglass Resistance	HGR	HGMX HGMY HGB HGSX HGSY
Strain	EPS	EPSXX1 EPSYY1 EPSZZ1 . . . EPSXY*ninteg* EPSYZ*ninteg* EPSZX*ninteg*

Table 1f Shells (3D)

Variable Aliases	Names	Component Names
Rate Deformation or RATEDFM	DOPT	DOPTXX1 DOPTYY1 DOPTZZ1 . . . DOPTXY*ninteg* DOPTYZ*ninteg* DOPTZX*ninteg*
Hourglass Energy	HGE	HGENGY
Pressure	PRESS	PRESS1 . . . PRESS*ninteg*
Von Mises	VONMIS	VONMIS1 . . . VONMIS*ninteg*
State Variables	SV	Components are the names provided in MATINT for each material.
Status	STATUS	STATUS

Table 1g **Rigid Shells (3D)**

Variable Aliases	Variable Names	Component Names
Element Mass	ELMASS	ELMASS
Status	STATUS	STATUS
Thickness	THICK	THICK
Element Bases	BASEL	BASELXX BASELYY BASELZZ BASELXY BASELYZ BASELZX

Table 1h SPH Elements (2D)

Variable Aliases	Variable Names	Component Names
Attributes	ATR	RADIUS VOL
Stress	SIG	SIGXX SIGYY SIGZZ SIGXY
Energy	ENERGY	ENERGY
Element Mass	ELMASS	ELMASS
Hourglass Resistance	HGR	HGX HGY
Strain	EPS	EPSXX EPSYY EPSZZ EPSXY
Stretch	STRECH	STRECHXX STRECHYY STRECHZZ STRECHXY
Rotation	ROTATE	COSTHETA SINTHETA
Density	RHO	DENSITY
Spin	SPIN	SPIN
	XLMIN	XLMIN
Viscous Pressure or Bulkq	VISPR	BULKQ
Rate Deformation or RATEDFM	DOPT	DOPTXX DOPTYY DOPTZZ DOPTXY
Hourglass Energy	HGE	HGENGY
Pressure	PRESS	PRESSURE
Von Mises	VONMIS	VONMISES

Table 1h SPH Elements (2D)

Variable Aliases	Variable Names	Component Names
Element Temperature	TEMN	TEMN (2D)
State Variables	SV	Components are the names provided in MATINT for each material.
EOS State	EOSSV	Components are the names created in EOSINT for each equation of state.
Status	STATUS	STATUS

Table 1i DMC Elements (2D)

Variable Aliases	Names	Component Names
Force	SFORCE	SFMX SFPX SFMY SFPY TENS
Radius	RAD	RAD
Angular Momentum	ANGMOM	ANGMOM
Angular Displacement	ANGROT	ANGROT
Angular Velocity	ANGVEL	ANGVEL
Angular Acceleration	ANGACC	ANGACC
Rotational Mass	RMOI	RMOI

Table 2 **Material Model State Variable Names**

Material Model	**State Variable Names**	
Rigid	None	
Elastic	None	
Elastic Plastic	ALPHA11 ALPHA22 ALPHA33 ALPHA12	ALPHA23 ALPHA31 EQPS
Viscoplastic	EQPS	SIGYLD
Damage	DAMAGE EVMAX FRAGSIZE	CRKDENS EQPS
Hydro	None	
Low Density Foam	PAIR	
Soil N Foams	EVMAX EVFRAC	EV NUM
EP Temp Depend	ALPHA11 ALPHA22 ALPHA33 ALPHA12	EQPS TEMP KAPPA
EP Hydrodynamic	ALPHA11 ALPHA22 ALPHA33 ALPHA12	ALPHA23 ALPHA31 RADIUS EQPS
EP Power Hardening	RADIUS	EQPS
Johnson Cook	RADIUS EQPS	TEMP ERATE
Hyperelastic	None	
Thorne Damage	DAMAGE RMAX EMAX	CRKDENS EQPS FRAGSIZE
Thermoelastic	TEMP SR ALPHA	YOUNGS YIELD
Wire Mesh	EVOL	YIELD

Table 2 **Material Model State Variable Names**

Material Model	State Variable Names	
Sandia Damage	ALPHAXX ALPHAYY ALPHAZZ ALPHAXY ALPHAYZ ALPHAXZ	K TEMP DTEMP/DT DAMAGE DDAM/DT
Orthotropic Crush	CRUSH	
PLH Strength	RADIUS EQPS TEARING	DECAY TDECAY
New Foam	PAIR	EVOL

Table 3 Material Model Required Material Cues

Material Model	Material Cues	
Rigid	Contact modulus	
Elastic	Youngs modulus	Poissons ratio
Elastic Plastic	Youngs modulus Poissons ratio Yield stress	Hardening modulus Beta
Viscoplastic	Youngs modulus Poissons ratio Yield stress	Hardening modulus Gamma P
Damage	Youngs modulus Poissons ratio Yield stress	M K Fracture toughness
Hydro	Pressure cuttoff	
Low Density Foam	Youngs modulus A B C	Nair Po Phi
Soil N Foams	Two mu Bulk modulus A0 A1	A2 Function id Pressure cutoff
EP Temp Depend	Youngs modulus Poissons ratio C1 C2 C3 C4 C4 C6 C7	C8 C9 C10 C11 C12 Beta Rhocv Temp
EP Hydrodynamic	Youngs modulus Poissons ratio Yield stress	Hardening modulus Beta Pressure cutoff
EP Power Hardening	Youngs modulus Poissons ratio Yield stress	Hardening modulus Hardening exponent Luders strain

Table 3 Material Model Required Material Cues

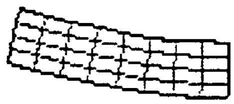
Material Model	Material Cues	
Johnson Cook	Youngs modulus Poissons ratio Yield stress Hardening modulus Hardening exponent	Rhocv Rate constant Thermal exponent Ref temperature Melt temperature
Hyperelastic	Function id	
Thorne Damage	Youngs modulus Poissons ratio K Fracture toughness Reload CP1	CP2 CP3 Y1 Y2 Y3 Rmin
Thermoelastic	Alpha function Youngs function Yield function	Poissons ratio Cp
Wire Mesh	Youngs modulus Poissons ratio A	B Tension
Sandia Damage	Youngs modulus Poissons ratio Initial temperature Thermal expansion Heat coeff C1 C2 C3 C4 C5 C6 C7 C8	C9 C10 C11 C12 C13 C14 C15 C16 C17 C18 Damage constant Initial damage
Orthotropic Crush	Compact Youngs modulus Compact Poissons ratio Compact Yield stress X id Y id Z id XY id YZ id	ZX id Full compaction Modulus X Modulus Y Modulus Z Modulus XY Modulus YZ Modulus ZX

Table 3 Material Model Required Material Cues

Material Model	Material Cues	
PLH Strength	Youngs modulus Poissons ratio Yield stress Hardening constant	Hardening exponent Luders strain Failure value Decay constant
New Foam	Youngs modulus Poissons ratio A B	Poly P0 PhiI

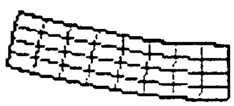
PRONTO3D Examples

Beam



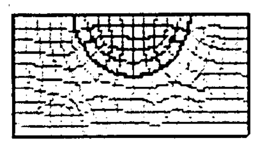
Cask Impacting Rail

Beam Restart

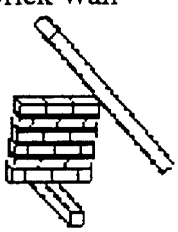


Restart

Contact Chatter

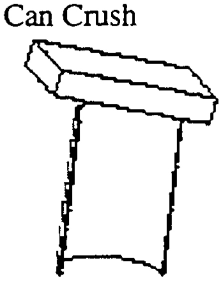


Brick Wall

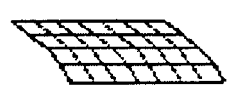


Shell Beam

Can Crush

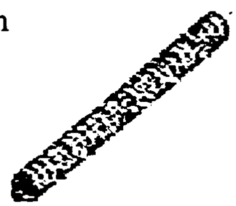


Shell Cylindrical Panel

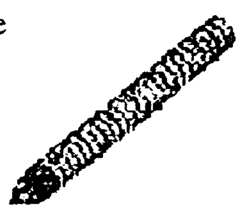


Shell Tearing

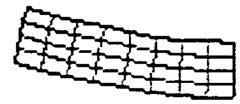
Cavity Expansion:
Aluminum



Cavity Expansion:
Concrete



Beam



Keywords beam-bending, hourglass control, pressure load

Description

This is a simple beam-bending example using a uniform pressure load, a symmetry plane, and a pinned support.

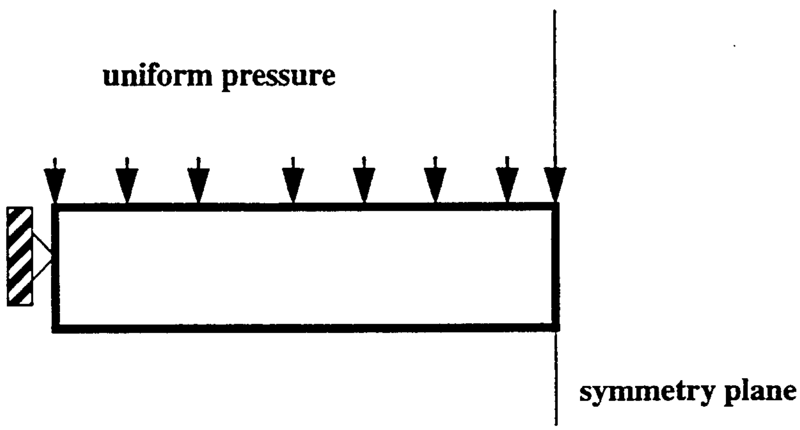


Figure 1 **Schematic of the example model**

The beam example problem is based on Flanagan and Belytschko's orthogonal hourglass control [Flanagan, D.P. and Belytschko, T., 1981]. This example is a severe problem for hourglassing as no deflection is possible without exciting the hourglass modes. The example is presented here to verifiy that the hourglass stiffness controls the hourglass modes. This simple problem tests the accuracy of the hourglass algorithm. It has only 32 elements, one side set for a pressure load, and one node set for a pinned boundary condition. The material model for this problem does not correspond to any real material.

Finite Element Model

The finite element model uses only 32 elements, one side set for the pressure load, and one node set for the no displacement boundary condition. A schematic of the mesh is shown in Figure 2.

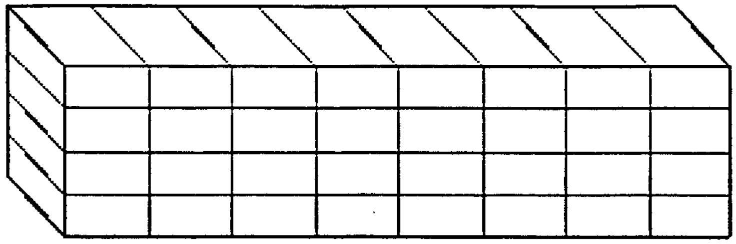


Figure 2 **Finite Element Model Mesh**

A plane strain assumption is created by prescribing no displacement boundary conditions along the front and back sides of the beam. The beam has a unit thickness so that it can be compared with results from PRONTO2D.

Results and Corroborative Data

Figure 3 shows a contour plot of the normal stress in the x-direction (σ_{xx}) at four different times (t = 0, 2, 4 and 6 msec). In this plot, the stress field is uniform in the center of the beam but is distorted by the pinned support. This contour plot is superimposed onto the deformed shape of the beam.

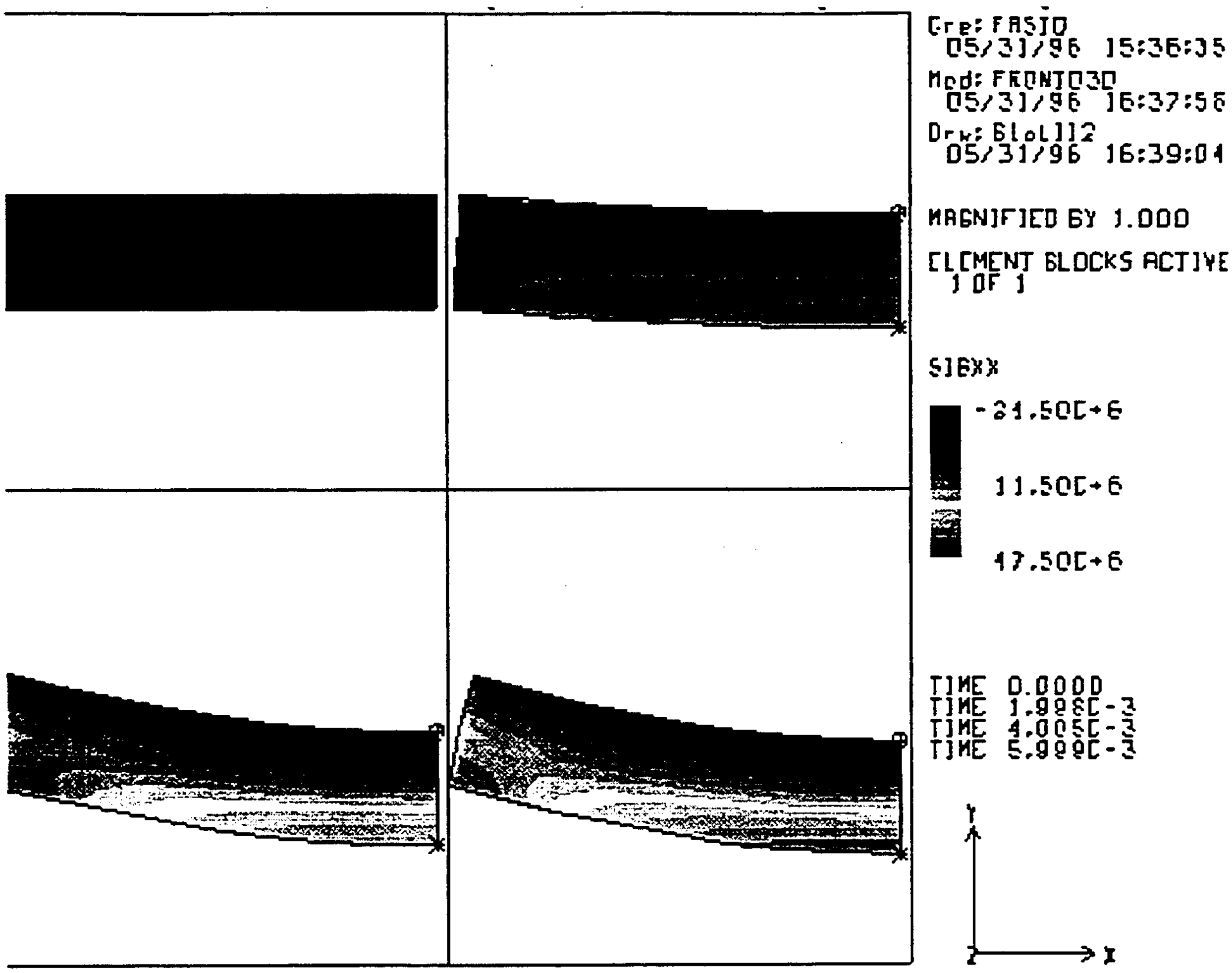


Figure 3 **Finite Element Analysis Results: Contour plot of stress in the x-direction at time t=0, 2, 4 and 6 msec.**

For this example the pressure load was applied using a step function at t = 0 sec. Under this loading condition, the beam will oscillate with time. Figure 4 shows a plot from a history file of the velocity of the beam at the node located at COORD = 0.3, 0.0, 0.0. This node moves up and down with time as is expected.

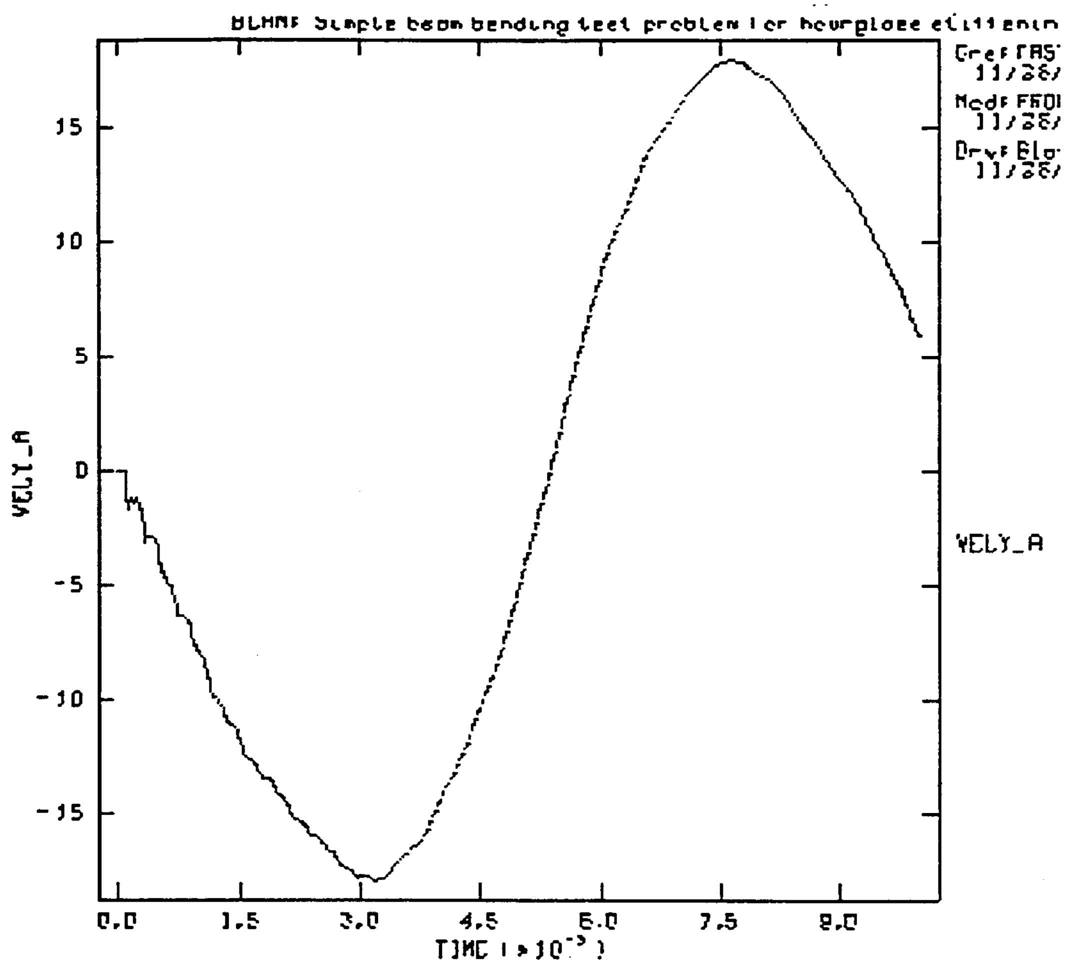


Figure 4 **Finite element results: Plot of velocity at COORD = 0.3, 0.0, 0.0. as a function of time.**

Figure 5 shows a plot of the kinetic energy as a function of time. As expected, the results oscillate with time.

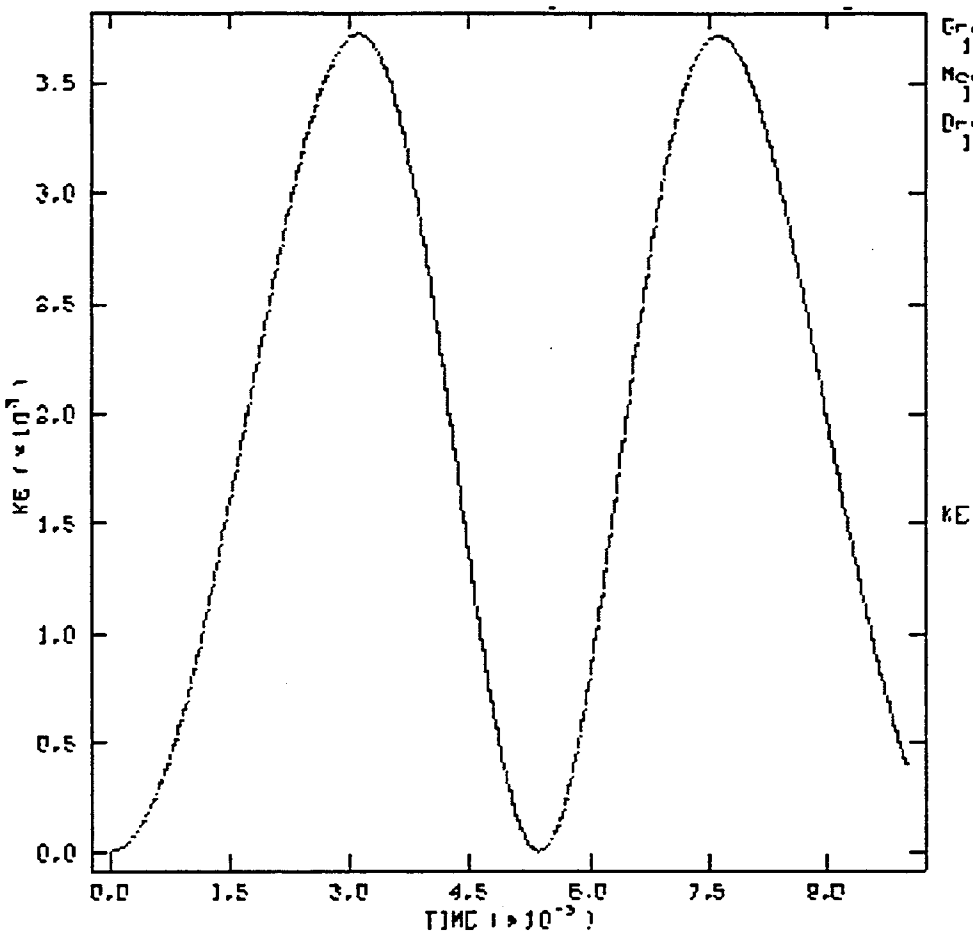


Figure 5 **Finite Element results: kinetic energy as a function of time.**

Figure 6 shows the deformed mesh plots when the hourglass stiffness is turned off. Without hourglass control, the elements distort. At time t = 2.0 msec the hourglass shape of the elements is visible. For time t = 6.0 msec, the hourglass distortions are so bad that the problem is not reconcilable.

Figure 7 shows the kinetic energy for the beam bending example when the hourglass stiffness is turned off.

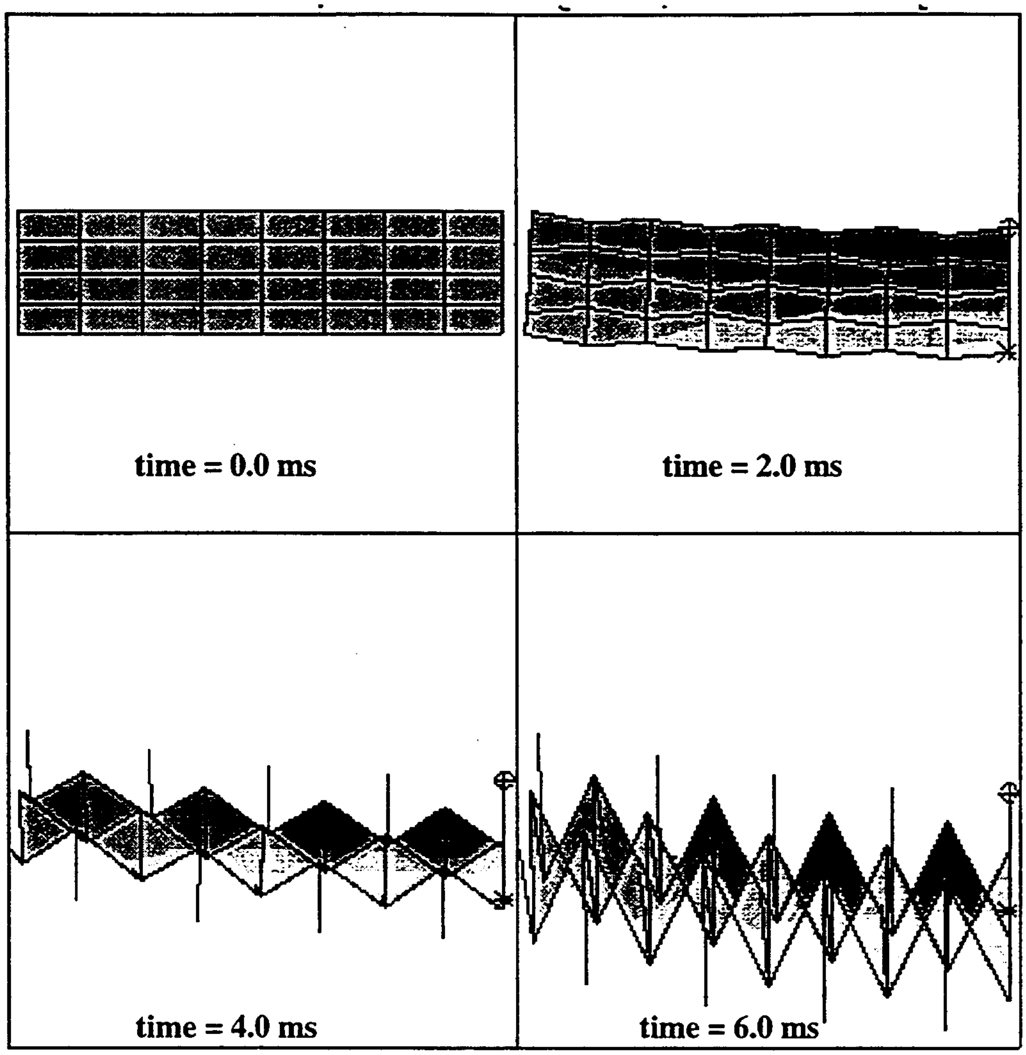


Figure 6 **Beam bending without hourglass control (not something you want to see in one of your analyses!).**

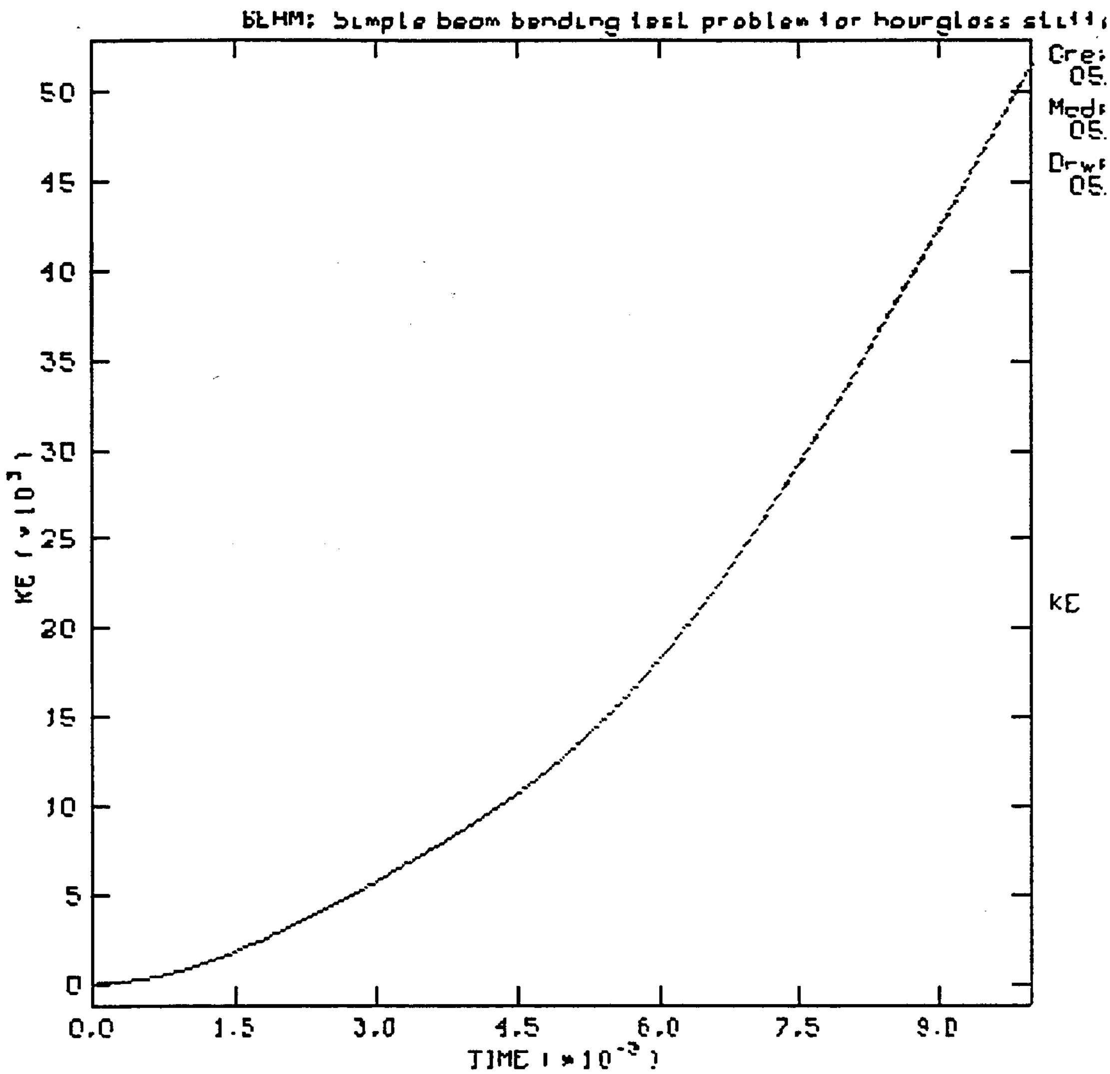


Figure 7 **Kinetic energy for beam bending problem for zero hourglass stiffness.**

Observations

The results for the default hourglass control have been compared with [Flanagan, D.P. and Belytschko, T., 1981] and with PRONTO2D. The results are in good agreement between PRONTO2D and PRONTO3D. As expected, when the hourglass stiffness is turned off, the hourglass modes of the elements are activated by the beam bending. The kinetic energy shown in Figure 7 does not oscillate with time as is shown in Figure 5. Instead, the kinetic energy grows exponentially.

Finite Element Input Data

beam.i

```
Title
   BEAM: Simple beam bending test problem for hourglass stiffening
```

```
Hourglass Stiffening = .05 0
Termination Time =  10.E-3
Output Time      = $ .25E-3
Plot Time        = .5E-3
Plot History, COORD = .4 0. 0., VARI=STRES, NAME=SIGA
Plot History, COORD = .2 0. 0., VARI=PRESS, NAME=PRESS
Plot History, COORD = .3 0. 0., VARI=VEL, NAME=A
Plot History, COORD = .4 0. 0., VARI=PRESS, NAME=PRESSA
No Displacement,X,5
No Displacement,Y,5
No Displacement,Z,100
No Displacement,Z,200
No Displacement,X,102
Pressure,103,1,720000
Function,1
   0,1
   1,1
Material,1,ELASTIC,1000.
   YOUNGS MODULUS = 1.E9 , POISSONS RATIO = 0
END
Plot Element = STRESS, HG
Exit
```

Problem Template

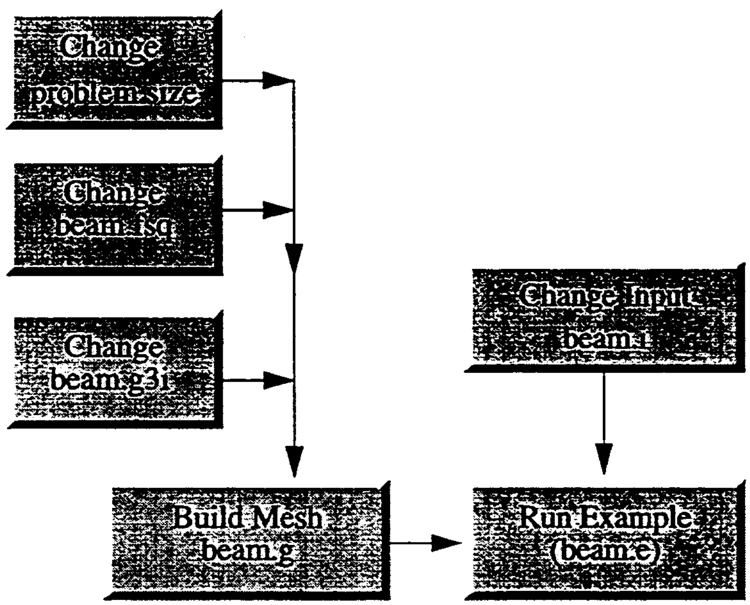


Figure 8 **Example template for building the mesh and running the example.**

Figure 8 shows an outline of how this problem is constructed using the SEACAS software system. Corresponding text files are included in the Mesh Generation and the Finite Element Input Data sections.

Mesh Generation

The mesh was generated using FASTQ and GEN3D. The following files were used:

```
proble.size - data file of APREPRO variables
beam.fsq - FASTQ input file
```

```
beam.g3i - GEN3D input file
```

The mesh can be made using the Makefile with the UNIX command:

```
make beam.g
```

problem.size

```
$ beam test problem
$ number of elements in x {ix = 8}
$ number of elements in y {iy = 4}
$ number of elelents in z {iz = 1}
$
$ number of processors x {px = 2}
$ number of processors y {py = 2}
$
```

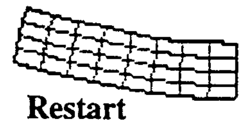
beam.fsq

```
$ {include(problem.size)}
POINT     1      0.000E+00      0.000E+00
POINBC    1     1
POINT     2      4.000E-01      0.000E+00
POINBC    2     2
POINT     3      4.000E-01      1.000E-01
POINBC    3     3
POINT     4      0.000E+00      1.000E-01
POINBC    4     4
POINT     5      0.000E+00      5.000E-02
POINBC    5     5
LINE      1   STR     1     2     0     {ix}     0.000000
LINEBC  101     1
LINE      2   STR     2     3     0     {iy}     0.000000
LINEBC  102     2
LINE      3   STR     3     4     0     {ix}     0.000000
SIDEBC  103     3
LINE      4   STR     4     5     0     {iy/2}     0.000000
LINEBC  104     4
LINE      5   STR     5     1     0     {iy/2}     0.000000
LINEBC  104     5
REGION    1     1    -1    -2    -3    -4    -5
EXIT
```

beam.g3i

```
$ {include(problem.size)}
translate {iz} 1
nsets front 100
nsets back 200
exit
```

Beam Restart



Keywords restart, beam-bending, hourglass control, pressure load

Description

This example shows how to write and read a restart using the beam-bending problem from Beam, which considers a simple beam-bending example using uniform pressure, a symmetry plane, and a pinned support. For details of the problem description and the finite element model, see Beam.

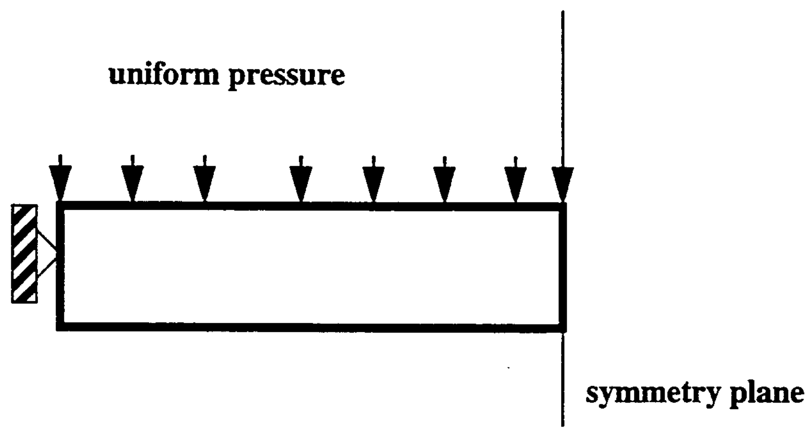


Figure 1 **Schematic of the example model**

Results and Corroborative Data

The problem runs in two parts. The first part runs for t = 5.0e-3 seconds and writes a restart file. The second part continues to 10.0e-3 seconds after reading the restart file. Figure 2 shows a plot of velocity versus time from the restarted job. The velocity should compare exactly with the velocity plot shown in Figure 4 from Beam. A plot of the kinetic energy for the restart file is shown in Figure 3.

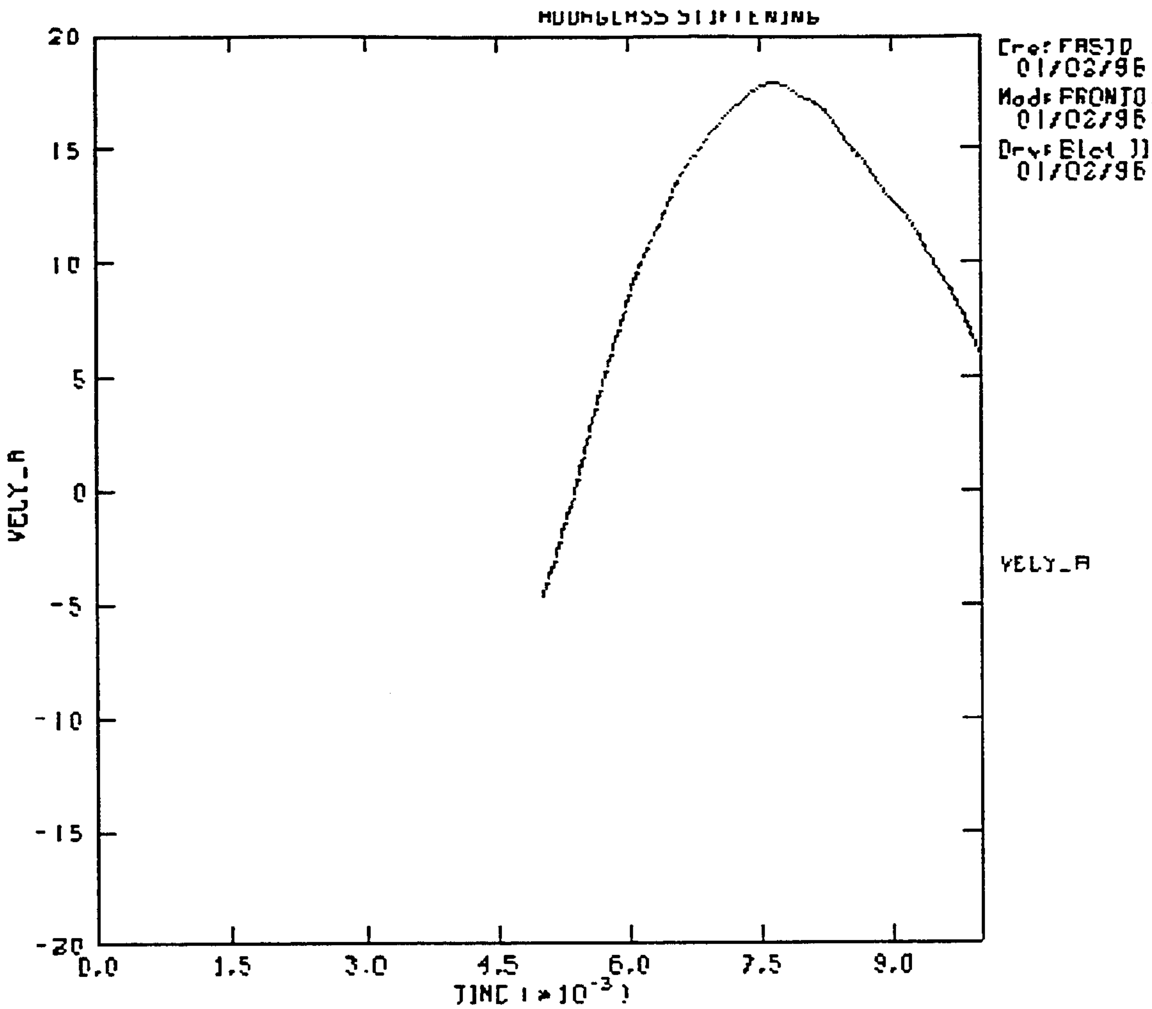


Figure 2 **Finite element results: Plot of velocity at COORD = 0.3, 0.0, 0.0. as a function of time.**

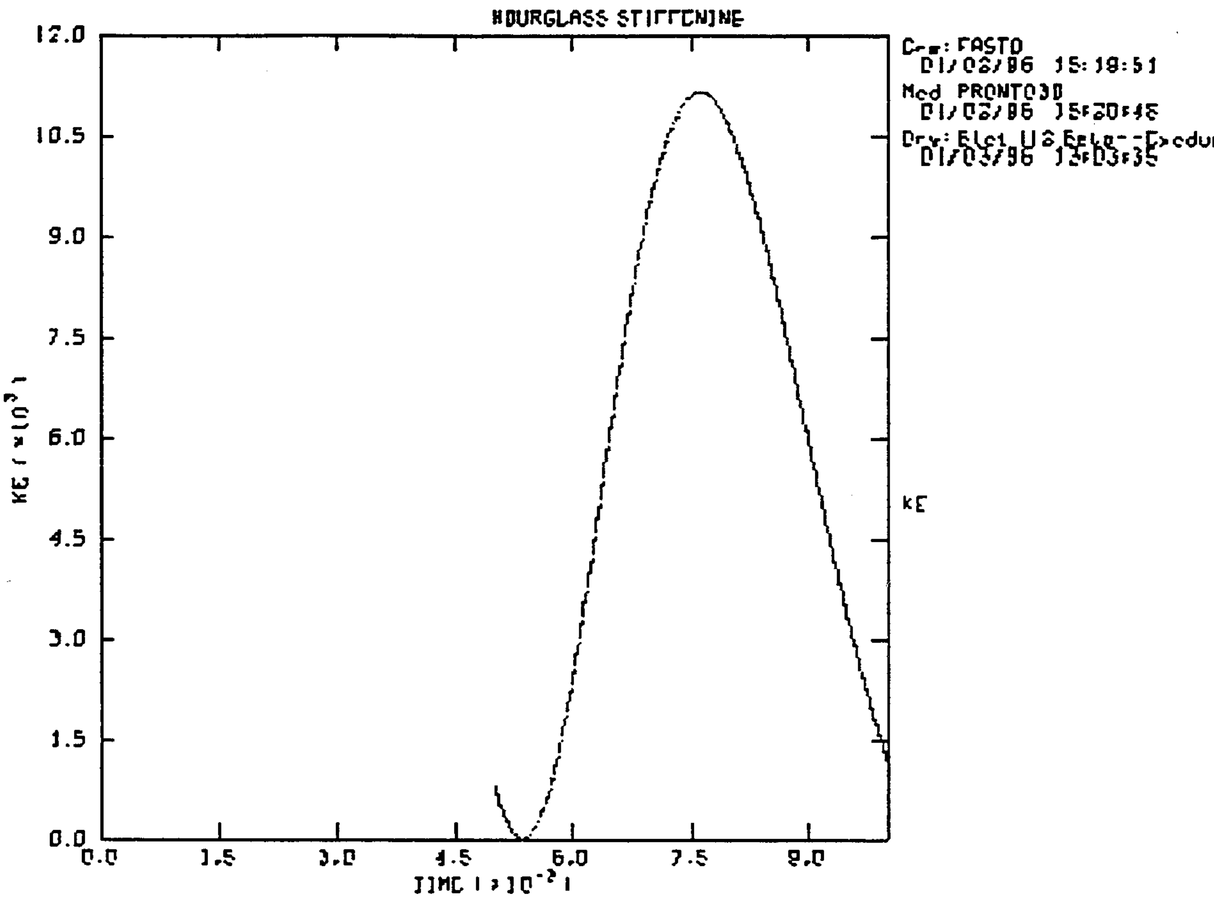


Figure 3 **Finite Element results: kinetic energy as a function of time.**

Observations

The restart results can be compared with the results from Beam. The kinetic energy and the velocity at the restart time are identical to the results shown in Beam.

Finite Element Input Data

In this case, input for the initial run and the restart are the same except for the termination times, read restart time, and the write restart time. The boundary conditions and loads could be changed at the time of restart. The number of element, connectivity, and the material types must be the same (sorry, you still cannot change lead to gold).

<u>Part 1 (beam.i from Beam)</u>

```
Title
BEAM: Simple beam bending test problem
Hourglass Stiffening = .05 0
Write Restart 2.5e-3
Termination Time =  10.e-3
Output Time      = $ .25E-3
Plot Time        = .5E-3
```

```
Plot History, COORD = .4, 0., 0. , VARI=STRES,NAME = SIGA
Plot History, COORD = .2 0. 0., VARI = PRESS, NAME = PRESS
Plot History, COORD = .3 0. 0., VARI = VEL, NAME = A
Plot History, COORD = .4 0. 0., VARI = PRESS, NAME = PRESSA
No Displacement,X,5
No Displacement,Y,5
No Displacement,Z,100
No Displacement,Z,200
No Displacement,X,102
Pressure,103,1,720000
Function,1
   0,1
   1,1
Material,1,ELASTIC,1000.
   YOUNGS MODULUS = 1.E9 , POISSONS RATIO = 0
End
Plot Element = STRESS,HG
Exit
```

Part 2 (beam_restart.i)

```
Title
   BEAM: Simple beam bending test problem (restart)
Hourglass Stiffening = .05 0
Read Restart 5.e-3
Termination Time =  10.e-3
Output Time      = $ .25E-3
Plot Time        = .5E-3
Plot History, COORD = .4, 0., 0. , VARI=STRES,NAME = SIGA
Plot History, COORD = .2 0. 0., VARI = PRESS, NAME = PRESS
Plot History, COORD = .3 0. 0., VARI = VEL, NAME = A
Plot History, COORD = .4 0. 0., VARI = PRESS, NAME = PRESSA
No Displacement,X,5
No Displacement,Y,5
No Displacement,Z,100
No Displacement,Z,200
No Displacement,X,102
Pressure,103,1,720000
Function,1
   0,1
   1,1
Material,1,ELASTIC,1000.
   YOUNGS MODULUS = 1.E9 , POISSONS RATIO = 0
End
Plot Element = STRESS,HG
Exit
```

Problem Template

Figure 4 shows an outline of how this problem is constructed using the SEACAS software system. Corresponding text files are included in the Finite Element Input Data section.

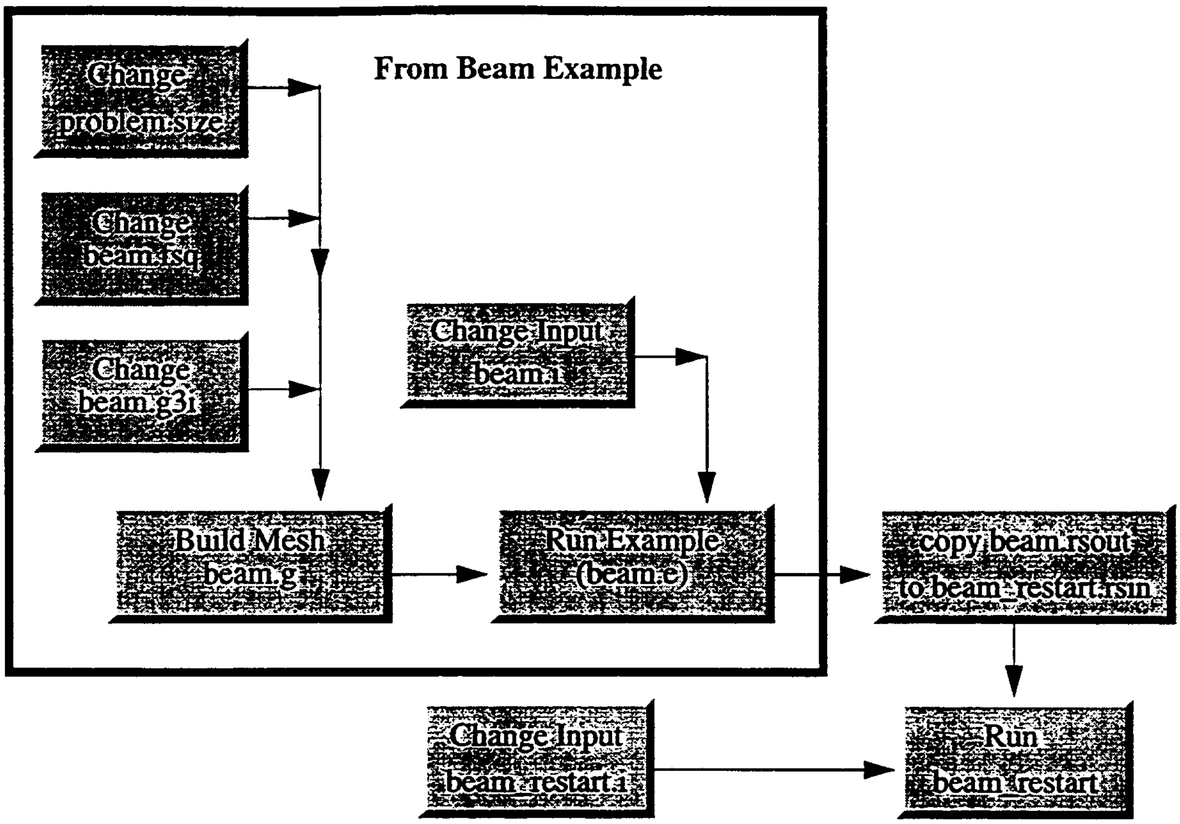


Figure 4 **Example template for building the mesh and running the example.**

After the initial run (as outlined in the Beam example description), a restart file with the extension ".rsout" will be written. This file should be copied or moved to a file with an extension of ".rsin" so that it can be read for the restart. One could edit the initial input file, changing the read and write restart times, and then rerun the problem. However, fewer errors will be generated if a new input file is created for each part of the problem.

The restart files are stored in the EXODUS data format and can be viewed with the same tools used for the plot files. The specified restart time must match within 5% of the time written on the restart data file, or an error will be generated.

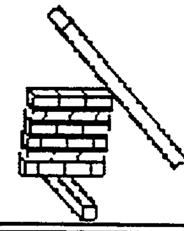
Mesh Generation

The mesh was generated for the Beam example (see the Mesh Generation section of that example).

The same GENESIS mesh can be read for both the initial and the restart file. Different mesh files can be used, however. The number of elements, number of nodes, and the connectivity must be the same in each mesh. However, the side sets and node sets used to define the kinematic boundary conditions and the loads can be changed.

Since the GENESIS file is stored with every EXODUS data file, the restart file, or the EXODUS file from the initial run, can be used as the mesh file. See the PRONTO on UNIX section for details of how to specify different names for the mesh, restart input, and restart output.

Brick Wall



Keywords global contact, initial velocity

Description

This example considers a wall of bricks being hit by an elastic-plastic rod. The initial geometry is shown in Figure 1. The impact causes the bricks to bounce off each other in an unpredictable manner. The Contact Material command is used to define the materials surfaces that are paired for contact.

One of the added capabilities of the contact material algorithm is the efficient modeling of multi-body impact without a-priori definition of contact surfaces. This example considers an elastic-plastic bar impacting a stack of 17 elastic bricks. A stationary elastic-plastic wall is also resting against the stack of bricks. All contact nodes and contact surfaces on the bodies were automatically defined using the contact material algorithm. For more information on the contact algorithm, see [Heinstein, M.W., Attaway, S.W., Mellow, F.J. and Swegle, J.W., 1993].

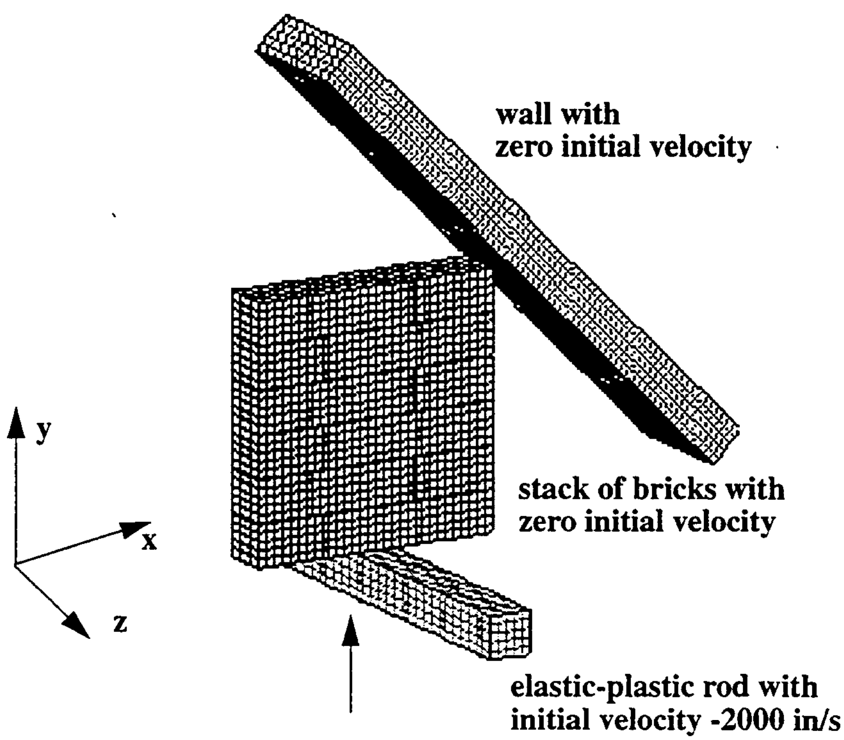


Figure 1 **Schematic of the example model**

Finite Element Model

The finite element mesh is shown in Figure 1. The number of elements per brick is adjustable. In addition, the number of bricks in the stack can be adjusted.

An initial velocity of v = 2000 in/sec was given to the impacting rod, using the initial velocity command. After t = 0, the rod flies free through space until contact is detected. The bricks were meshed with a small gap between each brick. If this small gap is not included, then a three-point contact will occur at the intersection of the bricks.

A global contact search is performed each time step to detect the contact between the rod, bricks, and wall. Zero friction was assumed.

Results and Corroborative Data

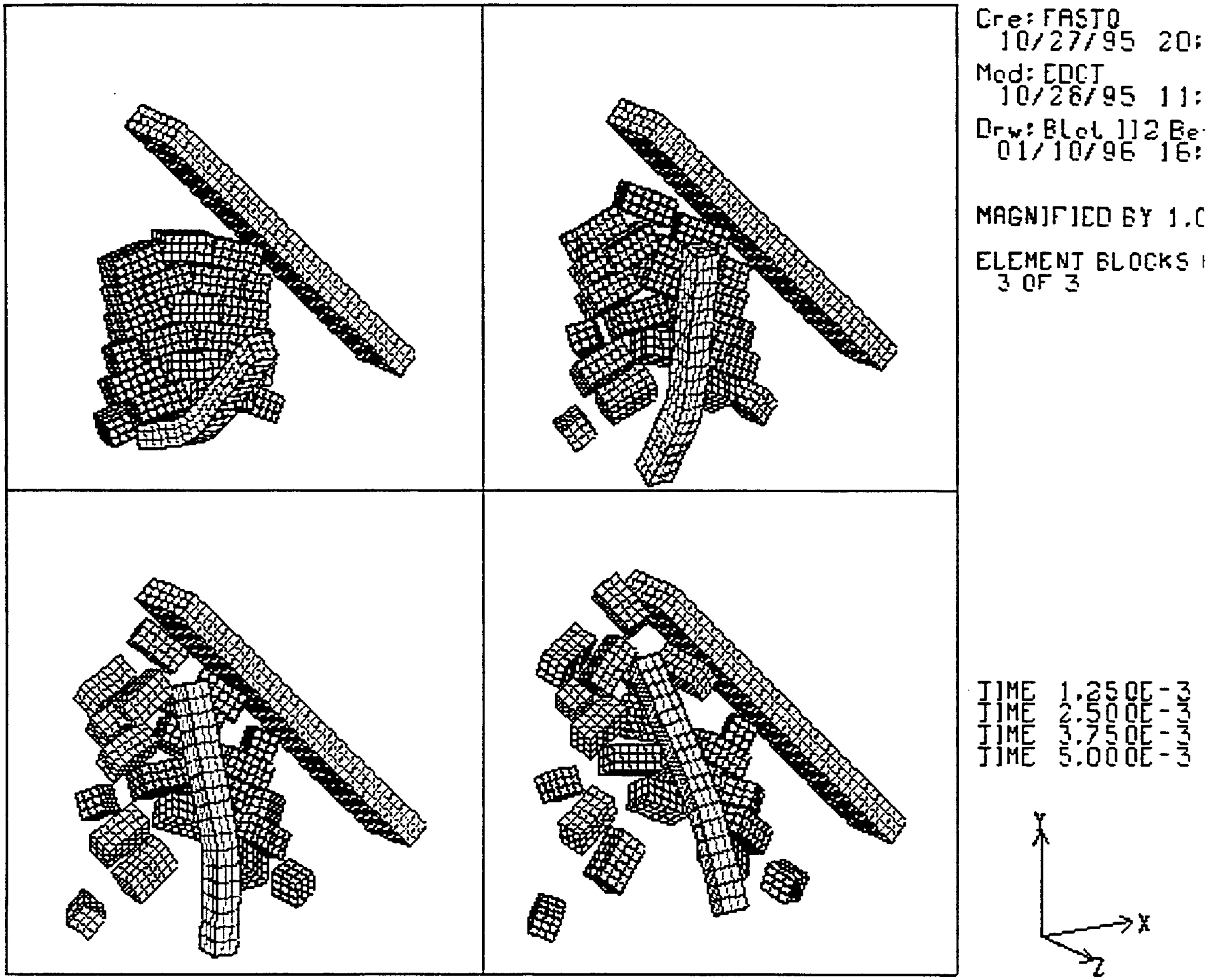


Figure 2 **Finite Element results: deformed shape at different times.**

Observations

For problems where random contact is anticipated, as in this example, each body could potentially impact any other body. For a contact-pairing algorithm, 19^2 contact pairs would be necessary, with each pair having $2n$ contact nodes. For the global contact searching algorithm, one search with $19n$ contact nodes is necessary. Assuming that each block has $n \approx 50$ contact nodes, 19^2 pairs would require $19^2(2n\log(2n)) = 239{,}843$ comparisons, whereas the global contact searching algorithm would require only $19n\log_2(19n) = 9397$ comparisons.

For this problem, the contact search takes less than 50% of the run time.

Finite Element Input Data

brick_wall.i

```
Title
  contact material test problem
Termination Time = .005
Plot Time        = .0001
Output Time      = .0001
Hourglass Stiffening .01 .03
$ rod
Material, 2, elastic plastic, .00074
  youngs modulus, 30e6
  poisons ratio, .3333
  hardening modulus 1000.
  beta = 1
  yield stress = 30000.
end
Material, 1, elastic , .00074
  youngs modulus, 30e6
  poisons ratio, .3333
end

$ wall
Material, 3, elastic plastic, .0005
  youngs modulus, 16e6
  poisons ratio, .3333
  hardening modulus 1000.
  beta = 1
  yield stress = 10000.
end
$ initial velocity for rod
Initial Velocity Material 2  0., 2000., 0.

$ no element data to be plotted
Plot Element =
Plot Nodal  displacement

$ include all materials in contact search
 Contact Material 1
 Contact Material 2
 Contact Material 3

Exit
```

Problem Template

Figure 3 shows an outline of how this problem is constructed using the SEACAS software system. Corresponding text files are included in the Mesh Generation and the Finite Element Input Data sections.

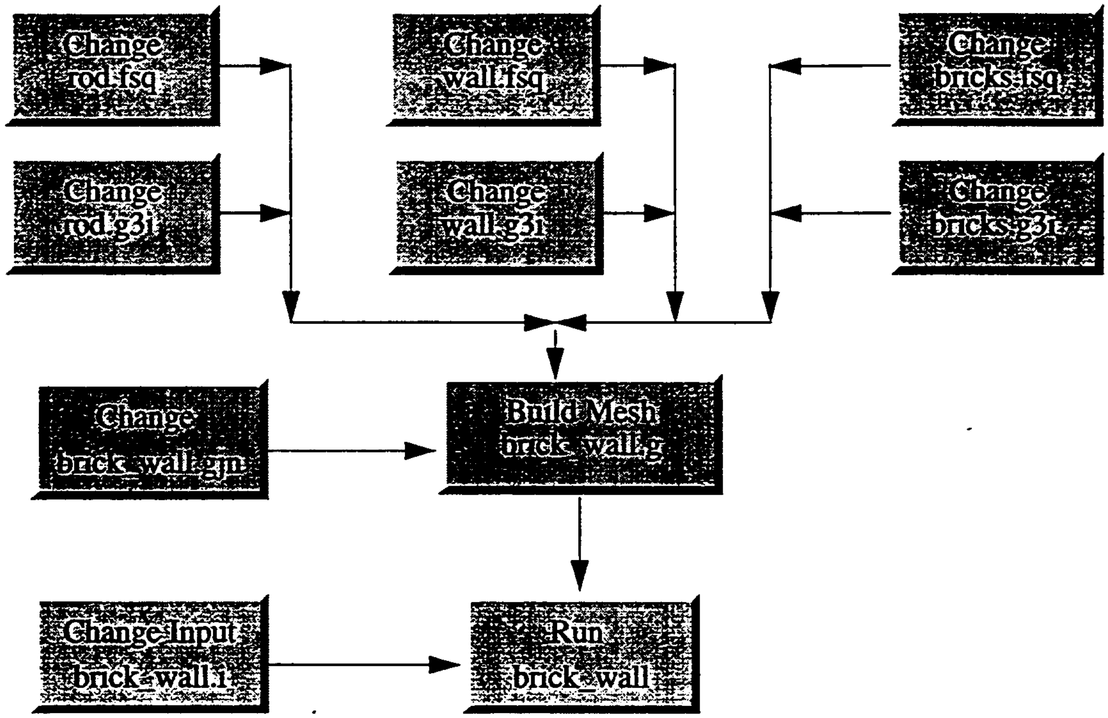


Figure 3 **Example template for building the mesh and running example.**

Mesh Generation

The mesh was generated using FASTQ, GEN3D, and GJOIN. The following files were used:

```
rod.fsq, bricks.fsq, wall.fsq - FASTQ input files
rod.g3i, bricks.g3i, wall.g3i - GEN3D input files
brick_wall.gjn - GJOIN input file
```

The mesh can be made using the Makefile with

```
make brick_wall.g
```

rod.fsq

```
title
 brick wall example
$ STEEL BLOCK

POINT 500 1.25 0.
POINT 501 1.75 0.
POINT 502 1.75 -.5
POINT 503 1.25 -.5

LINE 501 STR 500 501 0 5
LINE 502 STR 501 502 0 5
LINE 503 STR 502 503 0 5
LINE 504 STR 503 500 0 5

REGION 500 2 -501 -502 -503 -504


EXIT
```

bricks.fsq

```
TITLE
 STEEL BALL HITTING BRICKS
$ {e = .005}
POINT 1 {0.+e} {0.+e}
POINT 2 {1.-e} {0.+e}
POINT 3 {1.-e} {.5-e}
POINT 4 {0.+e} {.5-e}

LINE 1 STR 1 2 0 10
LINE 2 STR 2 3 0 5
LINE 3 STR 3 4 0 10
LINE 4 STR 4 1 0 5

REGION 1 1 -1 -2 -3 -4

SIDEBC 1 1 2 3 4

POINT 11 {1.+e} {0.+e}
POINT 12 {2.-e} {0.+e}
POINT 13 {2.-e} {.5-e}
POINT 14 {1.+e} {.5-e}

LINE 11 STR 11 12 0 10
LINE 12 STR 12 13 0 5
LINE 13 STR 13 14 0 10
LINE 14 STR 14 11 0 5

REGION 11 1 -11 -12 -13 -14

SIDEBC 11 11 12 13 14

POINT 21 {2.+e} {0.+e}
POINT 22 {3.-e} {0.+e}
POINT 23 {3.-e} {.5-e}
POINT 24 {2.+e} {.5-e}

LINE 21 STR 21 22 0 10
LINE 22 STR 22 23 0 5
LINE 23 STR 23 24 0 10
LINE 24 STR 24 21 0 5

REGION 21 1 -21 -22 -23 -24

SIDEBC 21 21 22 23 24

$ SECOND ROW

POINT 101 {0.+e} {.5+e}
POINT 102 {.5-e} {0.5+e}
POINT 103 {.5-e} {1.-e}
POINT 104 {0.+e} {1.-e}

LINE 101 STR 101 102 0 5
LINE 102 STR 102 103 0 5
LINE 103 STR 103 104 0 5
LINE 104 STR 104 101 0 5

REGION 101 1 -101 -102 -103 -104

SIDEBC 101 101 102 103 104

POINT 111 {.5+e} {.5+e}
POINT 112 {1.5-e} {.5+e}
```

```
POINT 113 {1.5-e} {1.-e}
POINT 114 {.5+e} {1.-e}

LINE 111 STR 111 112 0 10
LINE 112 STR 112 113 0 5
LINE 113 STR 113 114 0 10
LINE 114 STR 114 111 0 5

REGION 111 1 -111 -112 -113 -114

SIDEBC 111 111 112 113 114

POINT 121 {1.5+e} {.5+e}
POINT 122 {2.5-e} {0.5+e}
POINT 123 {2.5-e} {1.-e}
POINT 124 {1.5+e} {1.-e}

LINE 121 STR 121 122 0 10
LINE 122 STR 122 123 0 5
LINE 123 STR 123 124 0 10
LINE 124 STR 124 121 0 5

REGION 121 1 -121 -122 -123 -124

SIDEBC 121 121 122 123 124

POINT 131 {2.5+e} {.5+e}
POINT 132 {3.0-e} {0.5+e}
POINT 133 {3.0-e} {1.-e}
POINT 134 {2.5+e} {1.-e}

LINE 131 STR 131 132 0 5
LINE 132 STR 132 133 0 5
LINE 133 STR 133 134 0 5
LINE 134 STR 134 131 0 5

REGION 131 1 -131 -132 -133 -134

SIDEBC 131 131 132 133 134

$ THIRD ROW

POINT 201 {0.+e} {1.+e}
POINT 202 {1.-e} {1.+e}
POINT 203 {1.-e} {1.5-e}
POINT 204 {0.+e} {1.5-e}

LINE 201 STR 201 202 0 10
LINE 202 STR 202 203 0 5
LINE 203 STR 203 204 0 10
LINE 204 STR 204 201 0 5

REGION 201 1 -201 -202 -203 -204

SIDEBC 201 201 202 203 204

POINT 211 {1.+e} {1.+e}
POINT 212 {2.-e} {1.+e}
POINT 213 {2.-e} {1.5-e}
POINT 214 {1.+e} {1.5-e}

LINE 211 STR 211 212 0 10
LINE 212 STR 212 213 0 5
LINE 213 STR 213 214 0 10
```

```
LINE 214 STR 214 211 0 5

REGION 211 1 -211 -212 -213 -214

SIDEBC 211 211 212 213 214

POINT 221 {2.+e} {1. +e}
POINT 222 {3.-e} {1. +e}
POINT 223 {3.-e} {1.5-e}
POINT 224 {2.+e} {1.5-e}

LINE 221 STR 221 222 0 10
LINE 222 STR 222 223 0 5
LINE 223 STR 223 224 0 10
LINE 224 STR 224 221 0 5

REGION 221 1 -221 -222 -223 -224

SIDEBC 221 221 222 223 224


$ FOURTH ROW

POINT 301 {0.+e} {1.5+e}
POINT 302 {.5-e} {1.5+e}
POINT 303 {.5-e} {2.0-e}
POINT 304 {0.+e} {2.0-e}

LINE 301 STR 301 302 0 5
LINE 302 STR 302 303 0 5
LINE 303 STR 303 304 0 5
LINE 304 STR 304 301 0 5

REGION 301 1 -301 -302 -303 -304

SIDEBC 301 301 302 303 304

POINT 311 {.5+e}  {1.5+e}
POINT 312 {1.5-e} {1.5+e}
POINT 313 {1.5-e} {2.-e}
POINT 314 {.5+e}  {2.-e}

LINE 311 STR 311 312 0 10
LINE 312 STR 312 313 0 5
LINE 313 STR 313 314 0 10
LINE 314 STR 314 311 0 5

REGION 311 1 -311 -312 -313 -314

SIDEBC 311 311 312 313 314

POINT 321 {1.5+e} {1.5+e}
POINT 322 {2.5-e} {1.5+e}
POINT 323 {2.5-e} {2.-e}
POINT 324 {1.5+e} {2.-e}

LINE 321 STR 321 322 0 10
LINE 322 STR 322 323 0 5
LINE 323 STR 323 324 0 10
LINE 324 STR 324 321 0 5

REGION 321 1 -321 -322 -323 -324

SIDEBC 321 321 322 323 324
```

```
POINT 331 {2.5+e} {1.5+e}
POINT 332 {3.0-e} {1.5+e}
POINT 333 {3.0-e} {2.-e}
POINT 334 {2.5+e} {2.-e}

LINE 331 STR 331 332 0 5
LINE 332 STR 332 333 0 5
LINE 333 STR 333 334 0 5
LINE 334 STR 334 331 0 5

REGION 331 1 -331 -332 -333 -334

SIDEBC 331 331 332 333 334

$ FIFTH ROW

POINT 401 {0.+e} {2.+e}
POINT 402 {1.-e} {2. +e}
POINT 403 {1.-e} {2.5-e}
POINT 404 {0.+e} {2.5-e}

LINE 401 STR 401 402 0 10
LINE 402 STR 402 403 0 5
LINE 403 STR 403 404 0 10
LINE 404 STR 404 401 0 5

REGION 401 1 -401 -402 -403 -404

SIDEBC 401 401 402 403 404

POINT 411 {1.+e} {2.+e}
POINT 412 {2.-e} {2. +e}
POINT 413 {2.-e} {2.5-e}
POINT 414 {1.+e} {2.5-e}

LINE 411 STR 411 412 0 10
LINE 412 STR 412 413 0 5
LINE 413 STR 413 414 0 10
LINE 414 STR 414 411 0 5

REGION 411 1 -411 -412 -413 -414

SIDEBC 411 411 412 413 414

POINT 421 {2.+e} {2. +e}
POINT 422 {3.-e} {2. +e}
POINT 423 {3.-e} {2.5-e}
POINT 424 {2.+e} {2.5-e}

LINE 421 STR 421 422 0 10
LINE 422 STR 422 423 0 5
LINE 423 STR 423 424 0 10
LINE 424 STR 424 421 0 5

REGION 421 1 -421 -422 -423 -424

SIDEBC 421 421 422 423 424


EXIT
```

wall.fsq

```
title
  wall
point 1 5.5 0.
point 2 5.85355    .35355
point 4 .5  5.
point 3 .85355 5.35355

line 1 str 1 2 0 5
line 2 str 2 3 0 70
line 3 str 3 4 0 5
line 4 str 4 1 0 70

region 1 3 -1 -2 -3 -4

sidebc 600 1 2 3 4
exit
```

rod.g3i

```
translate 30 5
offset 0. 0. 3.
exit
```

bricks.g3i

```
translate 3 .6
ssets back 1000
ssets front 2000
exit
```

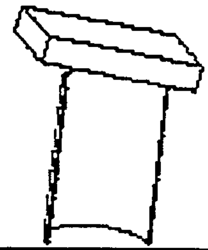
wall.g3i

```
translate 9 1.5
offset 0.05 0.05 .5
exit
```

brick_wall.gjn

```
bricks.g3d
wall.g3d
no
exit
yes
rod.g3d
no
exit
no
brick_wall.g
```

Can Crush



Keywords global contact, initial velocity, symmetry boundary conditions

Description

This example demonstrates the self-contacting capability of the contact detection algorithm [Heinstein, M.W., Attaway, S.W., Mellow, F.J. and Swegle, J.W., 1993]. This feature is important for modeling crash dynamics where buckling, tearing, and self contact is common. The elastic-plastic shell-like (can) structure shown in Figure 1 is impacted by an elastic-plastic plate. The can is 0.25 in. thick, has an inside radius of 5 in., and is 15 in. long. The bottom of the can is constrained in all directions. The 22x11 in. plate is 2.5 in. thick and is initially tilted at a 10° angle as it impacts the can at 5000 in/s.

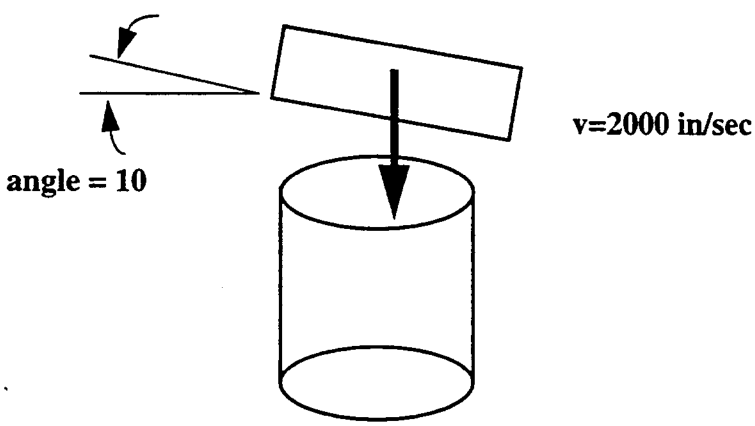


Figure 1 **Schematic of the example model**

Finite Element Model

The finite element mesh is shown in Figure 2. The mesh had 480 elements with 3 elements used through the thickness. Since the can will deform plastically, the number of integration points through the thickness is important. With only three integration points, the detection of plastic strain will be limited. More integration points would provide a more accurate integration of the plastic strain. The can is relatively thick, which allows the modeling of the through thickness transients with multiple hex elements. If the can were very thin, then using multiple elements through the thickness would require too small of a time step. For very thin structures, shell elements can be used to eliminate the through thickness time dependence.

A symmetry boundary condition was used along the center of the can and the block. A node set was used to identify the nodes that lie along the symmetry boundary condition. A No Displacement boundary condition was used to constrain the motion along the bottom of the can.

An initial velocity of v = 2000 in/sec was given to the block, using the Initial Velocity Material command. After t = 0, the block flies free through space until contact is detected.

A global contact search is performed each time step to detect the contact between the can and block. In addition, the global contact will check for self contact of the can as it buckles and folds back on itself. The contact can be specified two ways. The simplest is to use the Contact Material command. This option requires simply listing the materials included in the contact search. If no materials are listed, then all materials will be included in the global contact search. A second approach would be to define a contact surface only on the surfaces that are to be included in the contact search. To define contact surfaces, side sets must be generated at the time the mesh is created.

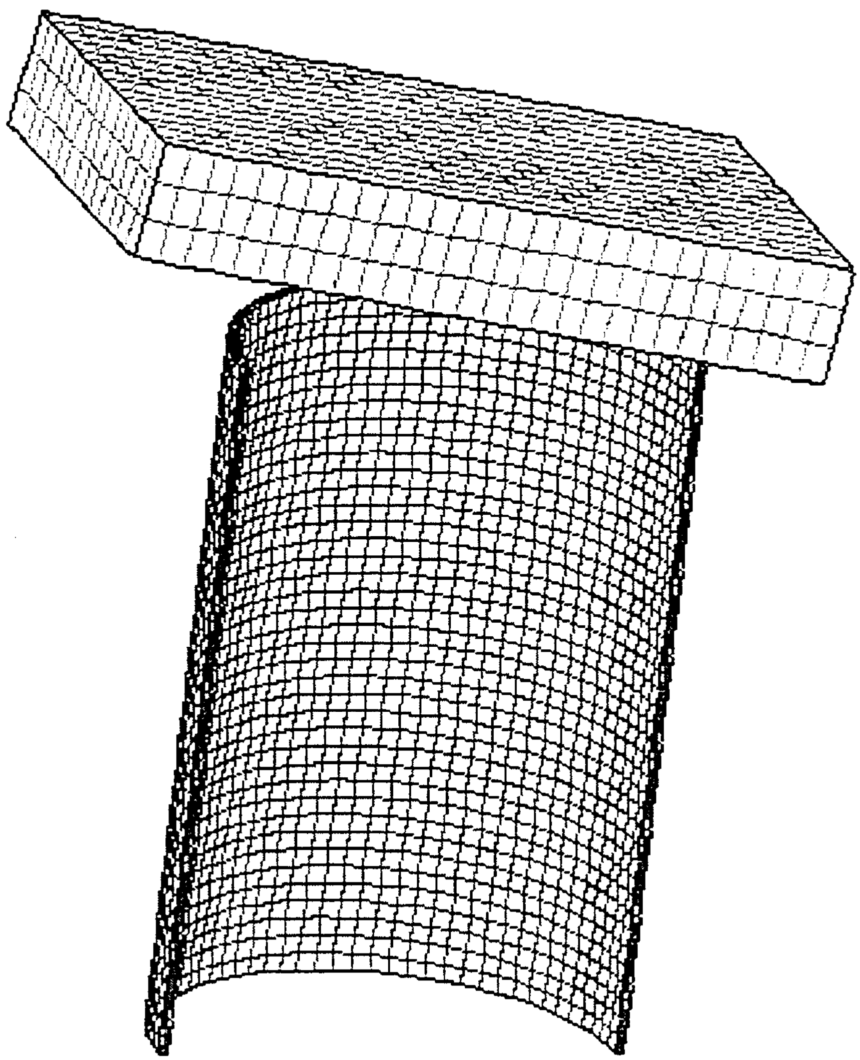


Figure 2 **Finite element mesh.**

Results and Corroborative Data

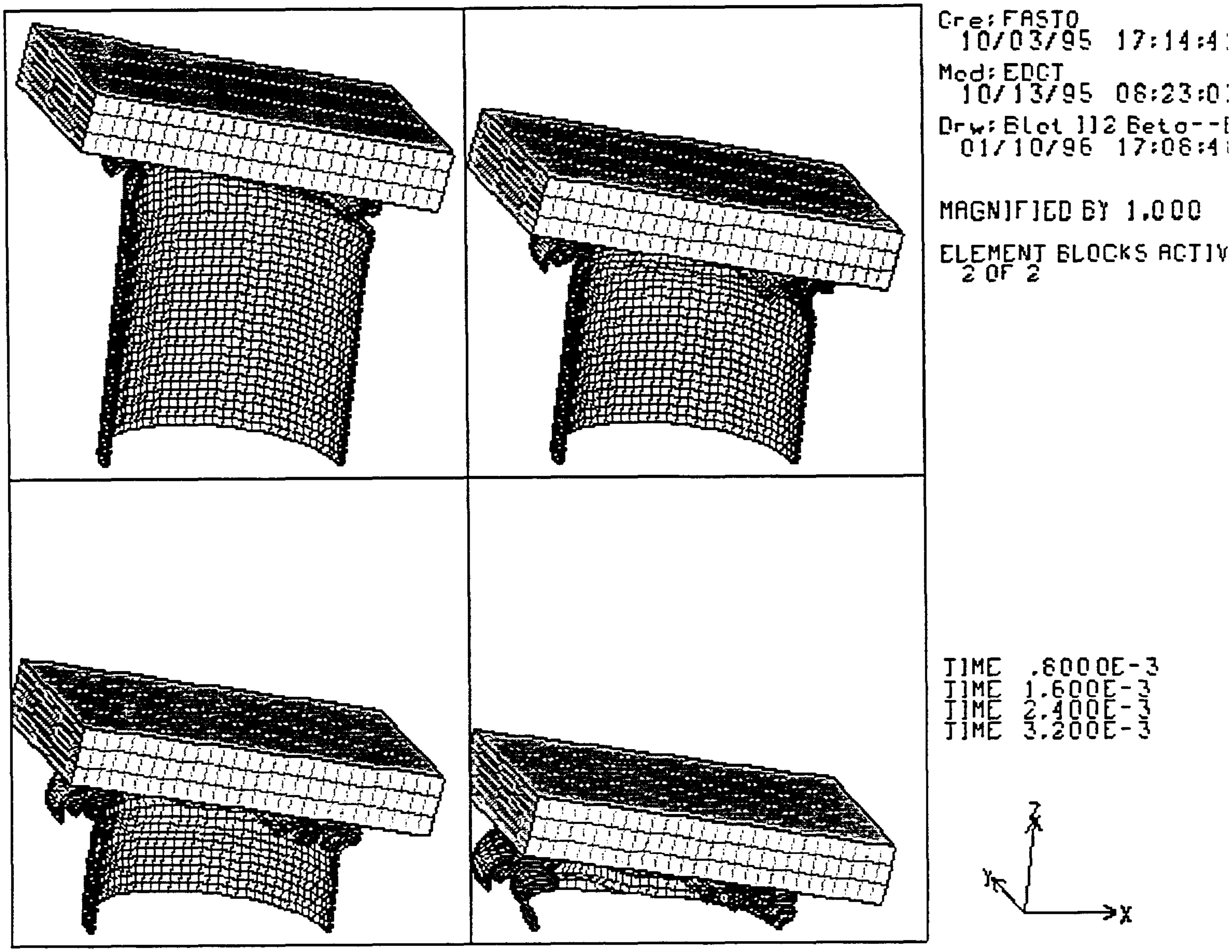


Figure 3 **Finite Element results: deformed shape at different times.**

Observations

While the results cannot be compared directly to experimental results (since friction was neglected), several observations can be made regarding the behavior of the contact algorithm. For example, interpenetration of the can by the block was prevented, which implies that the contact detection and contact correction are working correctly for this problem. If the can is viewed from the side, as shown in Figure 4, then a flat surface corresponding to the block is seen.

Complex self contact occurs as the can is crushed. The symmetry boundary conditions were correctly applied and interacted correctly with the contact detection and correction.

The initial velocity of the block was sufficient to more than crush the can. The No Displacement boundary condition applied along the bottom of the can only prevents motion of the bottom of the can. When the can buckles, the block is allowed to deform below the no displacement boundary condition in a nonphysical way. The calculation stops when an element inverts to give a negative element volume (we pushed way too hard). To insure that the can and block do not deform beyond the bottom of the can, a Rigid Surface could be used.

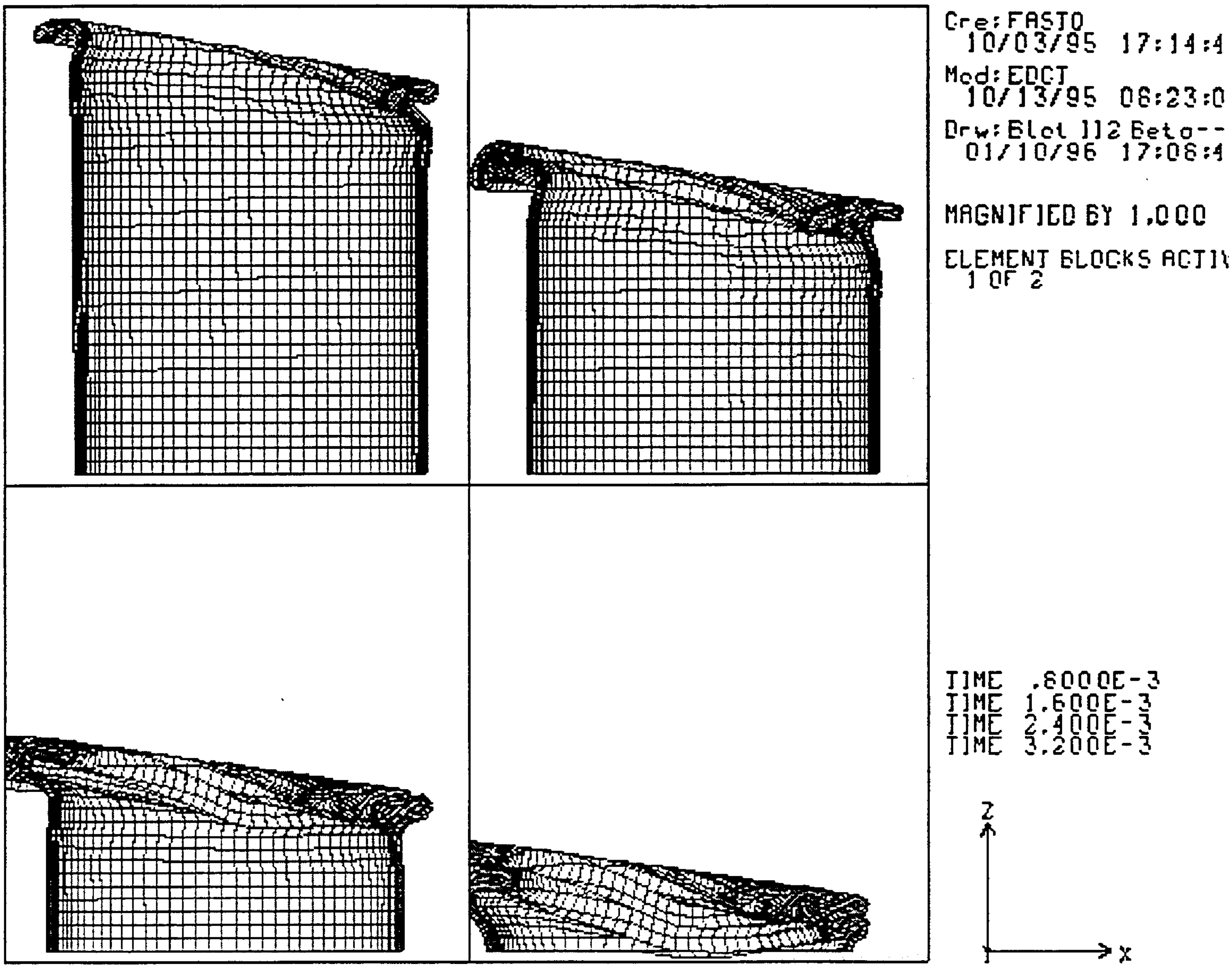


Figure 4 **Side view of can only.**

Finite Element Input Data

<u>**can_block_half.i**</u>

```
Title
```

```
   self contact test problem
Hourglass Stiffening .01 .03
Termination Time = .01
Plot Time   =  .0001
Output Time =  .0001
Material, 1, elastic plastic, .00074
   youngs modulus, 30e6
   poisons ratio, .3333
   hardening modulus 0.
   yields stress 30000
   beta = 1
end
Material, 2, elastic plastic, .00074
   youngs modulus, 30e6
   poisons ratio, .3333
   hardening modulus 0.
   yields stress 30000
   beta = 1
end

Contact Material 1
Contact Material 2

Initial Velocity Material 2  0., 0. -5000.

No Displacement z 100
No Displacement y 1

Plot Element =
Plot Nodal  displacement, velocity
Plot State = eqps

Exit
```

Problem Template

Figure 5 shows an outline of how this problem is constructed using the SEACAS software system. Corresponding text files are included in the Mesh Generation and the Finite Element Input Data sections.

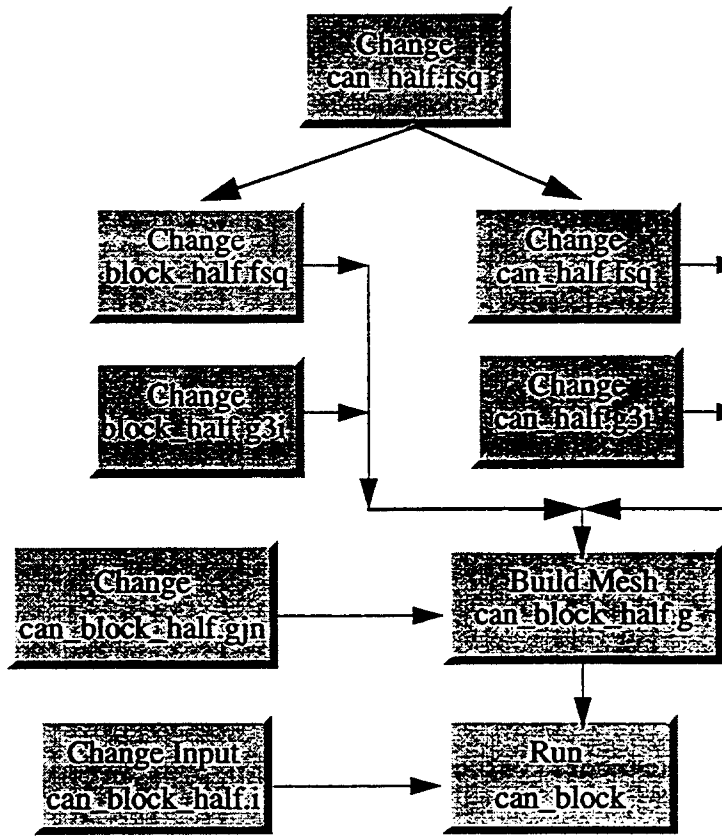


Figure 5 **Example template for building the mesh and running the example.**

Mesh Generation

The mesh was generated using FASTQ, GEN3D, and GJOIN. The following files were used:

```
can_half.fsq, block_half.fsq - FASTQ input files
can_half.g3i, block_half.g3i - GEN3D input files
can_block_half.gjn - GJOIN input file
```

The mesh can be made using the Makefile with

```
make can_block_half.g
```

The file can_half.size contains parameters for the mesh size and is included in each of the mesh files using APREPRO.

<u>can_half.size</u>

```
$ cylinder rad = {rad=5} t= {t=.2} int = {ithick=3} {irad=40}
$ length of cylender {cyllen=15}
$ number of elements in cylinder {icyl=40}
$ block  {e = 3}  int= {isq = 28}
$ number of elements in block thick {iblkt = 3 } block thickness {blkt=2.5}
$ block angle {angle=10}
$ number of processors {ix = 4} {iy=8}
```

can_half.fsq

```
title
 self contact test
{include(can_half.size)}

point 1 0 0
point 2 {rad} 0.
point 3 {rad+t} 0.
point 4 {-rad} 0.
point 5 {-rad-t} 0.

line 1 str 2 3 0 {ithick}
line 2 circ 2 4 1 {irad}
line 3 str 4 5 0 {ithick}
line 4 circ 3 5 1 {irad}

$ half can

region 1 1 -1 -2 -3 -4

nodebc 1   1 3

exit
```

block_half.fsq

```
title
 self contact test problem
{include(can_half.size)}
point 1 {rad+e}        0.
point 2 {-(rad+e)}     0.
point 3 {-(rad+e)} {rad+e}
point 4 {rad+e}    {rad+e}

line 1 str 1 2 0 {isq}
line 2 str 2 3 0 {isq/2}
line 3 str 3 4 0 {isq}
line 4 str 4 1 0 {isq/2}

region 1 2 -1 -2 -3 -4
nodebc 1 1
exit
```

can_half.g3i

```
{include(can_half.size)}
translate {icyl} {cyllen}
nodeset back  100
exit
```

block_half.g3i

```
{include(can_half.size)}
translate {iblkt} {blkt}
revolve y {180+angle}
revcen 0. 0. 0.
offset 0. 0.0 {tand(angle)*(rad+t)+.01}
exit
```

can_block_half.gjn

```
can_half.g3d
block_half.g3d
no
exit
finish
can_block_half.g
```

Cask Impacting Rail

Keywords contact, hourglass control

Description

In this example problem, a generic waste transportation cask is dropped from 30 feet onto a rigid rail. The impact velocity is 43.95 feet per second. The angle of impact is such that the center of gravity of the cask is over the corner where the impact occurs. The cask has 0.5 inch thick steel inner and outer liners with 3.5 inches of lead shielding between them. The analysis was run to a total time of 5 milliseconds.

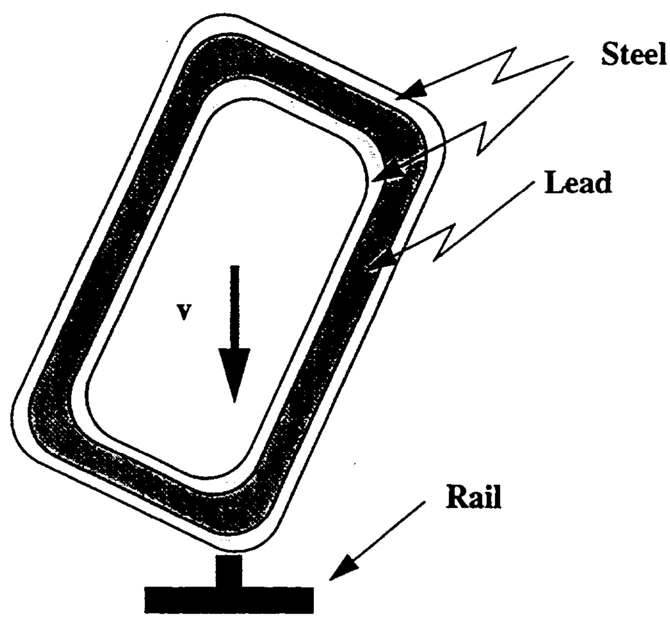


Figure 1 **Schematic of the example model**

Finite Element Model

The mesh for this analysis is shown in Figure 2. A symmetry boundary condition was used so that one half of the cask was modeled.

A Contact Surface was used to allow sliding between the lead and steel. Two types of contact surfaces can be used: paired contact surface, or global contact surface. The paired contact surface will generally require less cpu time; however, paired contacts require that the contact surfaces be defined using side sets. The global contact algorithm costs more; however, it takes much less set-up time because contact surfaces (side sets) do not have to be defined.

The lead was modeled as an elastic, perfectly plastic material. Note that the properties used here may not be the best properties for lead. The behavior of lead may include strain rate effects or more complex hardening behavior than assumed for this simple analysis.

Gravity was neglected in the analysis. The stresses generated by the initial gravity load will be very small relative to the stresses generated by the impact force. If the late time behavior of the cask is needed, such as a second bounce, then including gravity would ensure that the cask bounce was correct.

Typically casks are constructed by pouring lead in between the steel shells. As the lead becomes solid, cooling stresses would be generated. Here, it was assumed that these stresses would be small.

The cask closure system was not included in the analysis. The cask was treated as a symmetric body.

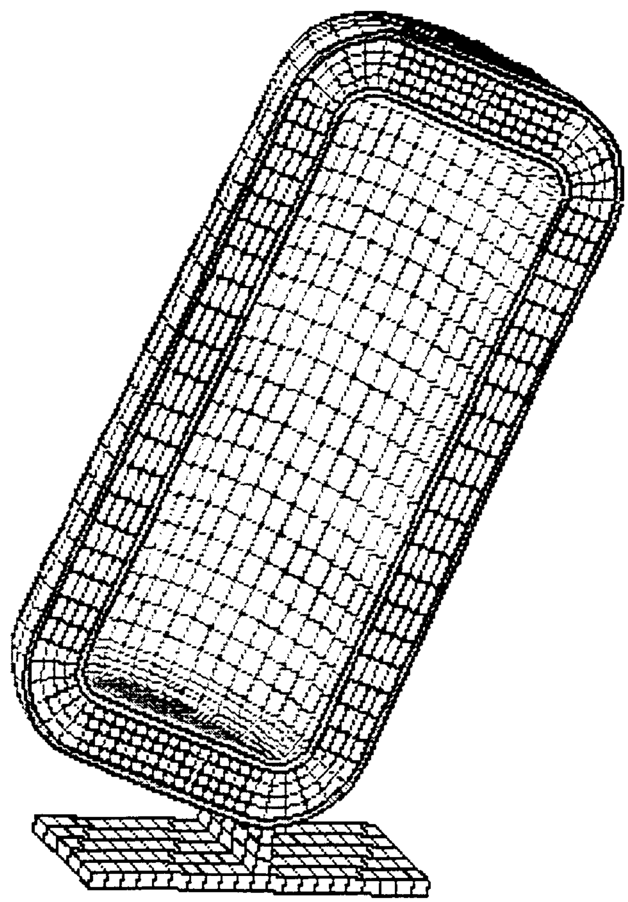


Figure 2 **Finite element mesh.**

Results and Observations

Figure 3 shows the total kinetic energy in the system. Rebound occurs at 4.6 milliseconds, at which time the deformations in the cask are the largest. As can be seen in Figure 4, the deformations in this analysis are not extremely large. Nevertheless, the materials in the cask, particularly the lead shielding, develop large plastic strains as shown in Figure 5.

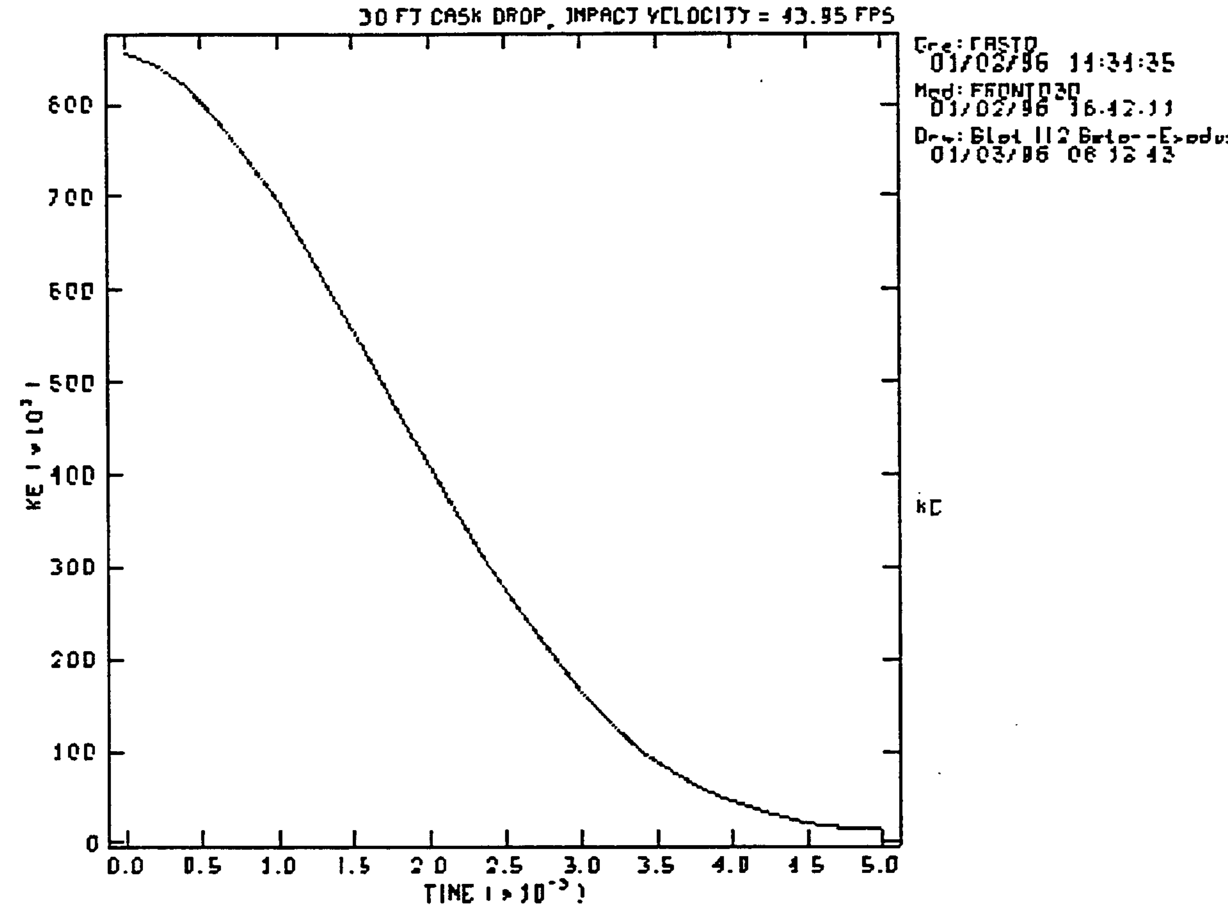


Figure 3 **Finite Element results: kinetic energy as a function of time.**

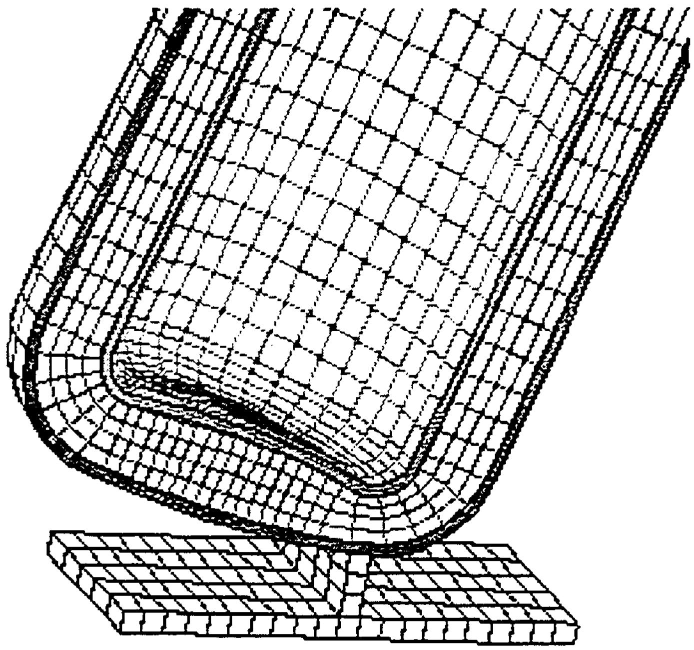


Figure 4 Deformed shape at 4.6 msec.

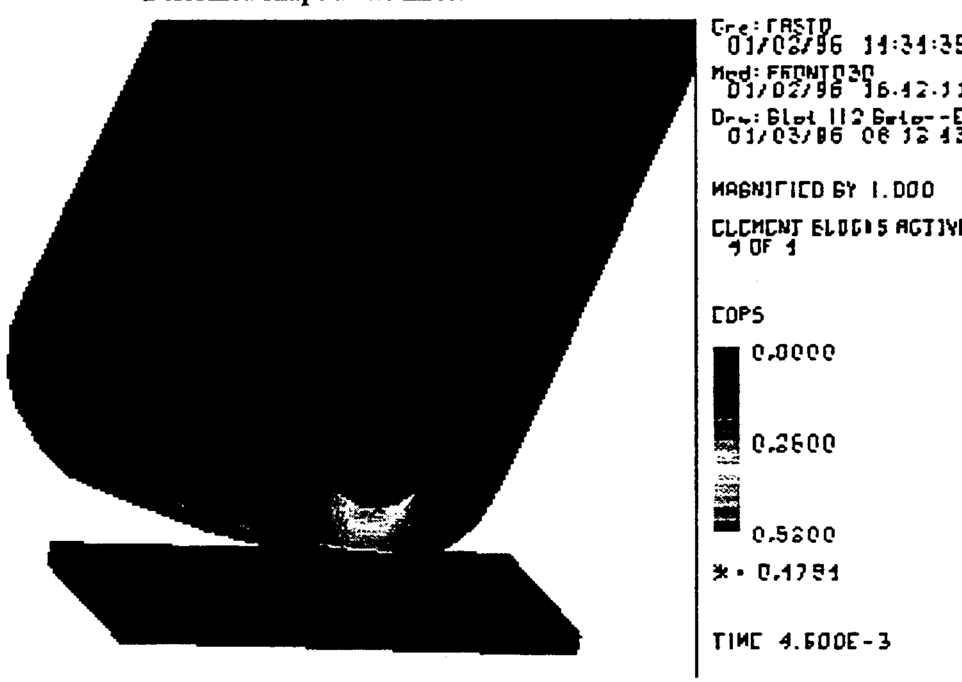


Figure 5 Plastic strain at 4.6 msec using hourglass stiffness.

This problem is included because it involves a large amount of contact data. A contact surface is defined between the liners and the shielding, and between the outer liner and the rail. These three contact surfaces make this a contact-intensive analysis.

Run Times

This problem involves a large amount of contact search. Additionally, the problem takes 10,932 time steps. Table I shows the total execution time for different machines. Different configurations were compared to illustrate the cost of each part of the algorithm.

Table I **Run times for cask-rail problem**

Machine	Operating system	Run time cpu sec	Microseconds per element cycle	Comments
CRAY X-MP 4/16	CTSS with CFTLIB	3732	32.7	Paired contact with FB hourglass control
CJ90	Unicos 8.1	5571	48.7	Paired contact with FB hourglass control
CJ90	Unicos 8.1	6563	53.9	Paired contact with assumed strain hourglass

Assumed Strain Hourglass versus F.B. Hourglass Control

Figure 6 shows a plot of the computed plastic strain at 4.6 msec when assumed strain hourglass is used. The Assumed Strain Hourglass command produces a much more flexible element that is better at capturing the perfectly plastic behavior of the lead. In this case, the assumed strain hourglass produced nearly twice the plastic strain as the F.B. hourglass control.

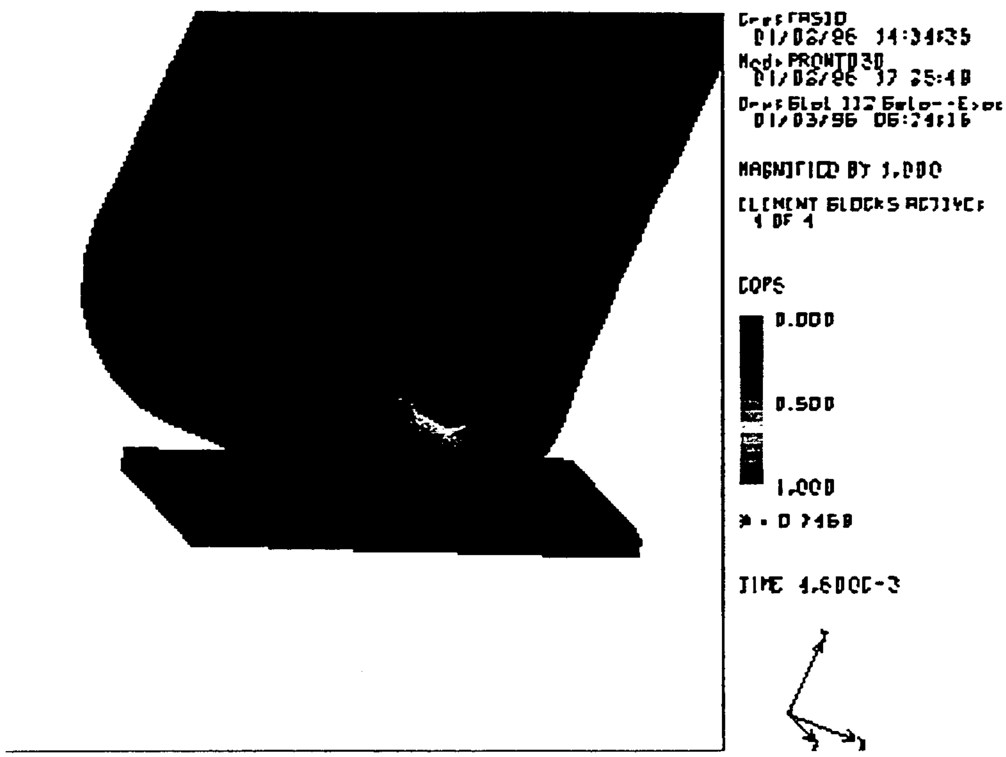


Figure 6 **Plastic strain at 4.6 msec computed using assumed strain hourglass.**

Finite Element Input Data

caskrail.i

```
Title
   30 FT CASK DROP, IMPACT VELOCITY = 43.95 FPS
Termination Time = 5.E-3
Plot Time = .2E-3
Output Time = .01E-3
Plot Nodal = DISPLACEMENT
Plot Element = VONMISES
Plot State = EQPS
Contact Surface = 202,201
Contact Surface = 102,101
Contact Surface = 88,89,0.,1
No Displacement Z = 1
No Displacement X = 3
No Displacement Y = 3
No Displacement Z = 3
Initial Velocity Material = 1 , 166.7 , -500.3 , 0.
Initial Velocity Material = 2 , 166.7 , -500.3 , 0.
Initial Velocity Material = 3 , 166.7 , -500.3 , 0.
Material 1 = ELASTIC PLASTIC , 7.366-4        $ STEEL
   YOUNGS MODULUS = 29.E6 , POISSONS RATIO = .33333
   YIELD STRESS = 40.E3 , HARDENING MODULUS = 40.E3
   BETA = 1.
End
Material 2 = ELASTIC PLASTIC , 10.53-4        $ LEAD
   YOUNGS MODULUS = 2.E6 , POISSONS RATIO = .44
   YIELD STRESS = 2000. , HARDENING MODULUS = 0. , BETA = 1.
End
Material 3 = ELASTIC PLASTIC , 7.366-4        $ STEEL
   YOUNGS MODULUS = 29.E6 , POISSONS RATIO = .33333
```

```
   YIELD STRESS = 40.E3 , HARDENING MODULUS = 40.E3
   BETA = 1.
End
Material 4 = ELASTIC , 1.        $ RIGID PLATE
   YOUNGS MODULUS = 1000. , POISSONS RATIO = 0.
End
Exit
```

The contact surfaces 202, 201, 102, and 101 are between the lead and steel. A friction factor of 0.1 was used with an equal kinematic partition. For the contact between the rail and the cask, surfaces 88 and 89 were used. Here the kinematic partition was set so that the rail was the master.

Problem Template

Figure 7 shows an outline of how this problem is constructed using the SEACAS software system. Corresponding text files are included in the Mesh Generation and the Finite Element Input Data sections.

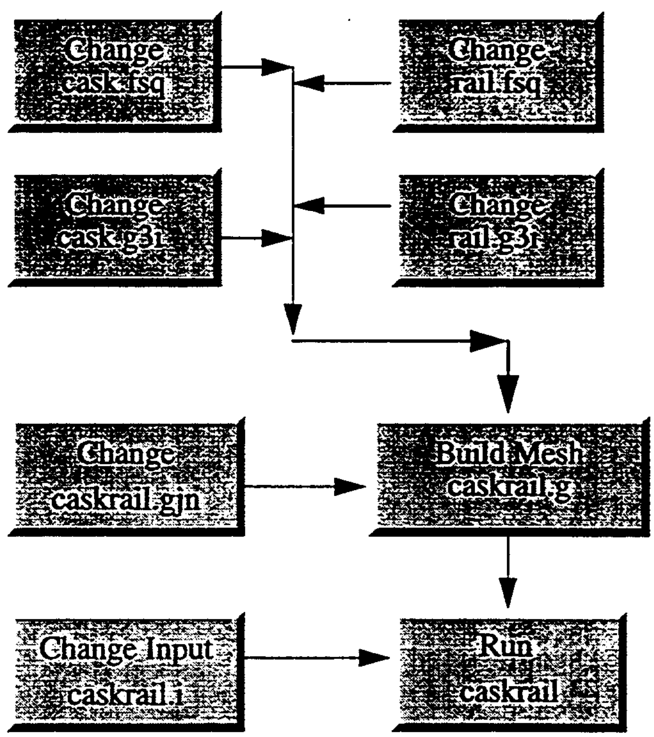


Figure 7 **Example template for building the mesh and running the example.**

Mesh Generation

The mesh was generated using FASTQ, GEN3D, and GJOIN. The following files were used:

```
cask.fsq, rail.fsq - FASTQ input files
cask.g3i, rail.g3i - GEN3D input files
caskrail.gjn - GJOIN input file
```

The mesh can be made using the Makefile with

```
make caskrail.g
```

cask.fsq

```
point  1 0 -20
point  2 6 -20
point  3 8 -18
point  4 8  18
point  5 6  20
point  6 0  20
point  7 6 -18
point  8 6  18
point  9 0 -20.5
point 10 6 -20.5
point 11 8.5 -18
point 12 8.5  18
point 13 6  20.5
point 14 0  20.5
point 15 8   0
point 16 8.5   0
line 1 str 1 2 0 8
line 2 circ 2 3 7 6
line 3 str 3 15 0 10 1.05
line 33 str 4 15 0 10 1.05
line 4 circ 4 5 8 6
line 5 str 5 6 0 8
line 6 str 9 10 0 8
line 7 circ 10 11 7 6
line 8 str 11 16 0 10 1.05
line 9 str 12 16 0 10 1.05
line 10 circ 12 13 8 6
line 11 str 13 14 0 8
line 12 str 1 9 0 4
line 13 str 2 10 0 4
line 14 str 3 11 0 4
line 15 str 4 12 0 4
line 16 str 5 13 0 4
line 17 str 6 14 0 4
side 2 8 9
side 3 3 33
sidebc 202 6 7 8 9 10 11
point  101 0.000 -20.501
point  102 6.001 -20.501
point  103 8.501 -18.001
point  104 8.501  18.001
point  105 6.001  20.501
point  106 0.001  20.501
point  107 6.001 -18.001
point  108 6.001  18.001
point  109 0.000 -23.50
point  110 6.00 -23.50
point  111 11.50 -18.00
point  112 11.50  18.00
point  113 6.00  23.50
point  114 0.00  23.50
point  115 8.501   0
point  116 11.50   0
line 101 str 101 102 0 8
line 102 circ 102 103 107 6
line 103 str 103 115 0 10 1.05
line 133 str 104 115 0 10 1.05
line 104 circ 104 105 108 6
line 105 str 105 106 0 8
line 106 str 109 110 0 8
line 107 circ 110 111 107 6
line 108 str 111 116 0 10 1.05
line 109 str 112 116 0 10 1.05
```

```
line 110 circ 112 113 108 6
line 111 str 113 114 0 8
line 112 str 101 109 0 4
line 113 str 102 110 0 4
line 114 str 103 111 0 4
line 115 str 104 112 0 4
line 116 str 105 113 0 4
line 117 str 106 114 0 4
side 102 108 109
side 103 103 133
sidebc 201 101 102 103 133 104 105
sidebc 102 106 107 108 109 110 111
point  201 0 -23.501
point  202 6.001 -23.501
point  203 11.501 -18.001
point  204 11.501  18.001
point  205 6.001  23.501
point  206 0  23.501
point  207 6.001 -18.001
point  208 6.001  18.001
point  209 0 -24.
point  210 6 -24.
point  211 12 -18.
point  212 12  18.
point  213 6  24.
point  214 0  24.
point  215 11.501   0
point  216 12   0
line 201 str 201 202 0 8
line 202 circ 202 203 207 6
line 203 str 203 215 0 10 1.05
line 233 str 204 215 0 10 1.05
line 204 circ 204 205 208 6
line 205 str 205 206 0 8
line 206 str 209 210 0 8
line 207 circ 210 211 207 6
line 208 str 211 216 0 10 1.05
line 209 str 212 216 0 10 1.05
line 210 circ 212 213 208 6
line 211 str 213 214 0 8
line 212 str 201 209 0 4
line 213 str 202 210 0 4
line 214 str 203 211 0 4
line 215 str 204 212 0 4
line 216 str 205 213 0 4
line 217 str 206 214 0 4
side 202 208 209
side 203 203 233
sidebc 101 201 202 203 233 204 205
sidebc 88  206 207 208 209 210 211
region 1 2 -112 -106 -113 -101
region 2 2 -113 -107 -114 -102
region 3 2 -114  102 -115  103
region 4 2 -115 -110 -116 -104
region 5 2 -116 -111 -117 -105
region 6 3 -12 -6 -13 -1
region 7 3 -13 -7 -14 -2
region 8 3 -14  2 -15  3
region 9 3 -15 -10 -16 -4
region 10 3 -16 -11 -17 -5
region 11 1 -212 -206 -213 -201
region 12 1 -213 -207 -214 -202
region 13 1 -214  202 -215  203
region 14 1 -215 -210 -216 -204
```

```
region 15 1 -216 -211 -217 -205
exit
```

rail.fsq

```
point 1 -.5 0
point 2  .5 0
point 3 -11 -3
point 4 -.5 -3
point 5  .5 -3
point 6 11 -3
point 7 -11 -4
point 8  11 -4
line 1 str 1 2 0 1
line 2 str 1 4 0 3
line 3 str 2 5 0 3
line 4 str 3 4 0 10
line 5 str 4 5 0 1
line 6 str 5 6 0 10
line 7 str 3 7 0 1
line 8 str 6 8 0 1
line 9 str 7 8 0 21
side 1 4 5 6
sidebc 89 6 3 1 2 4
linebc 3 1 2 3 4 5 6 7 8 9
region 1 4 -1 -3 -5 -2
region 2 4 1 -8 -9 -7
exit
```

cask.g3i

```
rotate 20 180.
center 1 2 3
nodeset front 1
nodeset back 11
exit
```

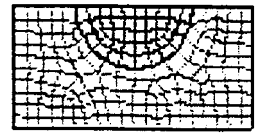
rail.g3i

```
translate 5 10.
revcen 0. 0. 0.
revolve z 18.43
offset 7.897 -23.702 0.
exit
```

caskrail.gjn

```
rail.g3d
cask.g3d
NO
nsets
combine 1 11
exit
FINISH
caskrail.g
```

Contact Chatter



Keywords contact, contact chatter, pressure load

Description

One of the difficulties with two curved surfaces contacting each other under high normal loads is the accurate determination of the push-back direction. The example shown in Figure 1 has a semicircular rod that is pushed into a semicircular cavity. A pressure load is applied to the flat surface of the rod. The pressure is ramped up over time, then held fixed.

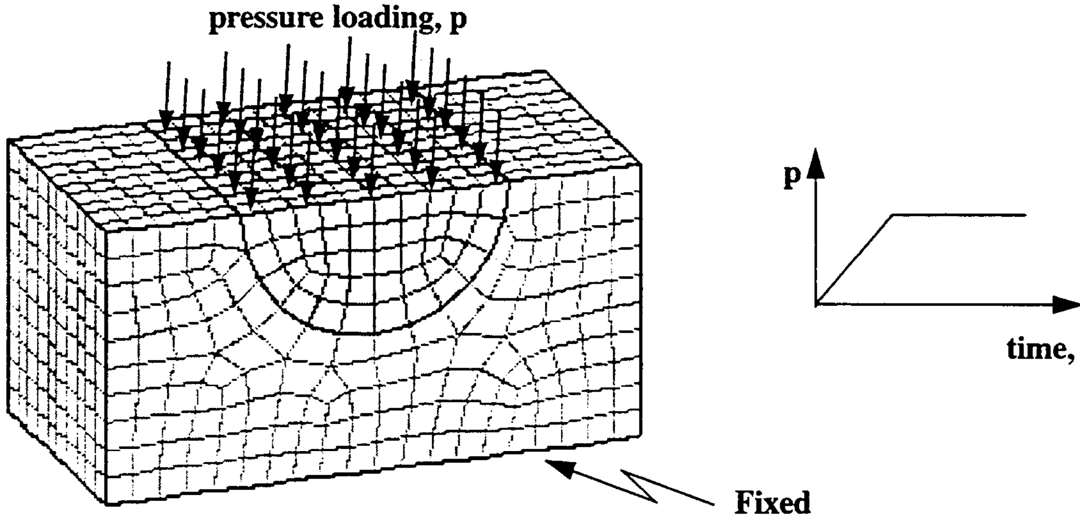


Figure 1 **Schematic of the example model**

Finite Element Model

The mesh for this analysis is shown in Figure 1. All the nodes on each contact surface are initially aligned with each other. As the pressure is ramped up, the contact nodes on the semicircle (convex surface) must be pushed back to the vertices of the contact surfaces on the cavity (concave surface).

The discrete nature of the surfaces makes it difficult to model the contacts correctly. The analyst often wants to model a smooth surface. However, with low order elements, the surfaces will have flat sides. The smoothness of the surface can be improved if more elements are used.

Here two different contact algorithms will be compared: the paired contact algorithm [Taylor, L.M. and Flanagan, D.P., 1989], and the global contact algorithm [Heinstein, M.W., Attaway, S.W., Mellow, F.J. and Swegle, J.W., 1993].

Results and Observations

The kinetic energy for the paired contact algorithm is shown in Figure 2. The deformed shape at time t = 2.5 msec is shown in Figure 3 with the displacements magnified to illustrate the excess hourglass energy.

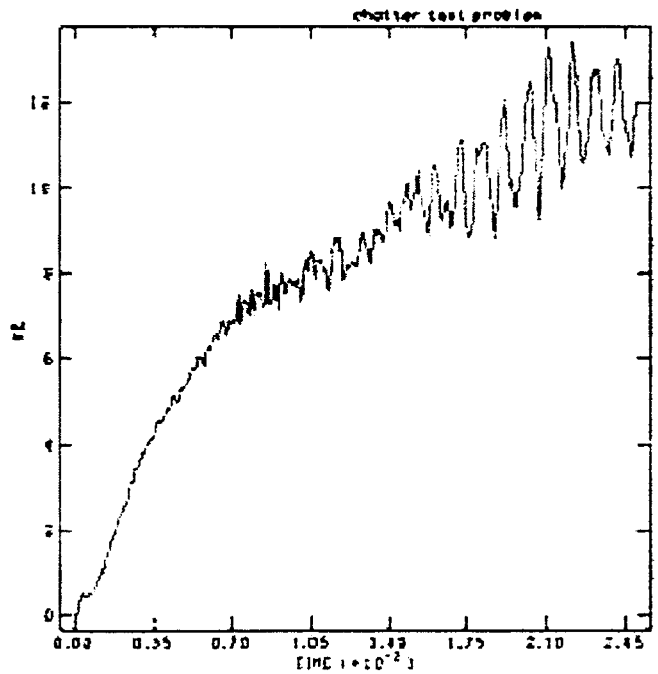


Figure 2 **The paired contact algorithm incorrectly determined the pushback direction and introduced noise (contact chatter) into the solution, resulting in the increase in kinetic energy shown here.**

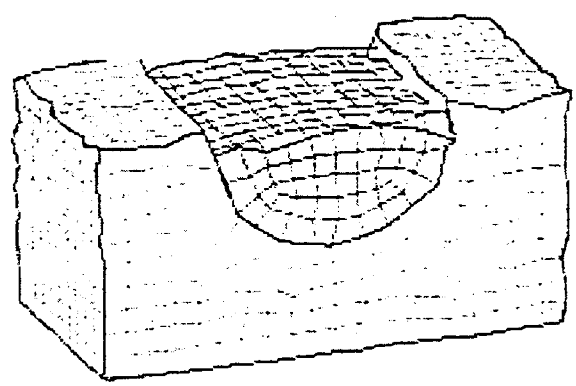


Figure 3 **Mesh hourglassing generated by contact chatter when the paired contact algorithm was used.**

The paired contact surface algorithm incorrectly determined the pushback direction and introduced noise (contact chatter) into the solution. The algorithm actually cannot decide which surface the contact node should be pushed back to, so it oscillates between the two surfaces. This eventually accumulates over many time steps and results in mesh hourglassing and increased kinetic energy, as shown in Figure 2 and Figure 3. (Hint: Try changing the default hourglass stiffness and hourglass viscosity for this problem to see the effect that these parameters have on the stability of the solution.)

The global contact detection algorithm (contact material or unpaired contact surface) correctly determines the pushback direction and does not introduce noise (see Figure 5 and Figure 6). Instead of trying to decide which surface the node should be pushed to, the global contact algorithm will push the node to the vertex of the two surfaces.

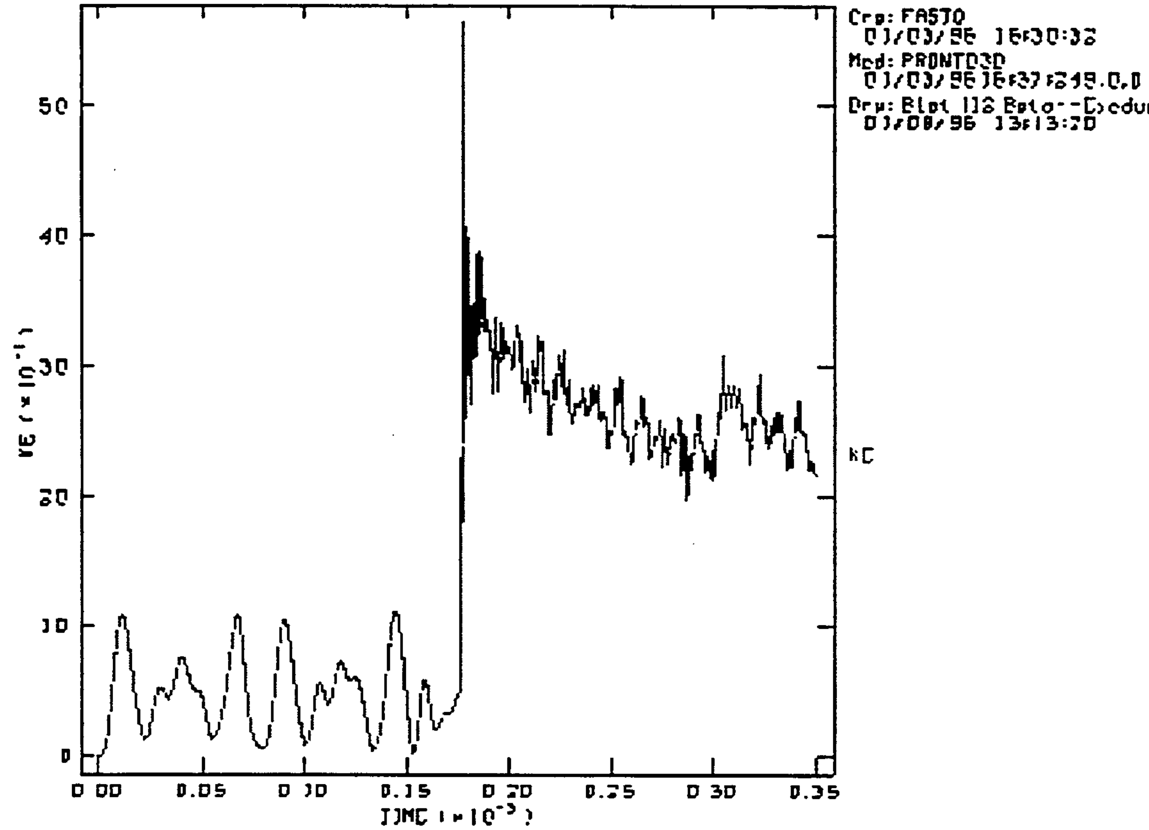


Figure 4 **Kinetic energy history using the General-Purpose Contact Detection Algorithm.**

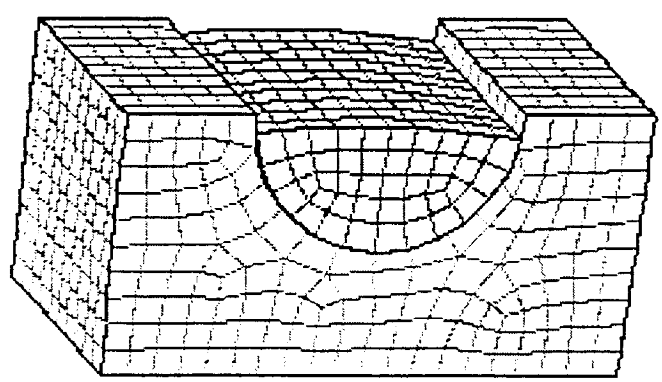


Figure 5 **Deformed shape (magnified 100 times) using General-Purpose Contact Detection Algorithm.**

Finite Element Input Data

chatter.i

```
Title
   chatter test problem (global contact)
Termination Time .00035
Plot Time   .00001
Output Time .00001
Write Restart .001

Plot History node=562 variable=vel name=nd562

Material 1 elastic .00074
   youngs modulus 30e6
   poissons ratio .3
End

Material 2 elastic .00074
   youngs modulus 30e6
   poissons ratio .3
End

Contact Surface 1
Contact Surface 2

Pressure 100 50 1.
Function 50
   0.        0.
   .00015 20000.
   .01   20000.
End

No Displacement y 11
No Displacement x 11

Exit
```

chatter_paired.i

```
Title
   chatter test problem (paired contact)
Termination Time .00035
Plot Time .00001
Output Time .00001
Write Restart .001

Plot History node=562 variable=vel name=nd562

Material 1 elastic .00074
   youngs modulus 30e6
   poissons ratio .3
end

Material 2 elastic .00074
   youngs modulus 30e6
   poissons ratio .3
end

Contact Surface 1 2

Pressure 100 50 1.
Function 50
```

```
    0.        0.
    .00015 20000.
    .01    20000.
end

No Displacement y 11
No Displacement x 11

Exit
```

Problem Template

Figure 6 shows an outline of how this problem is constructed using the SEACAS software system. Corresponding text files are included in the Mesh Generation and the Finite Element Input Data sections.

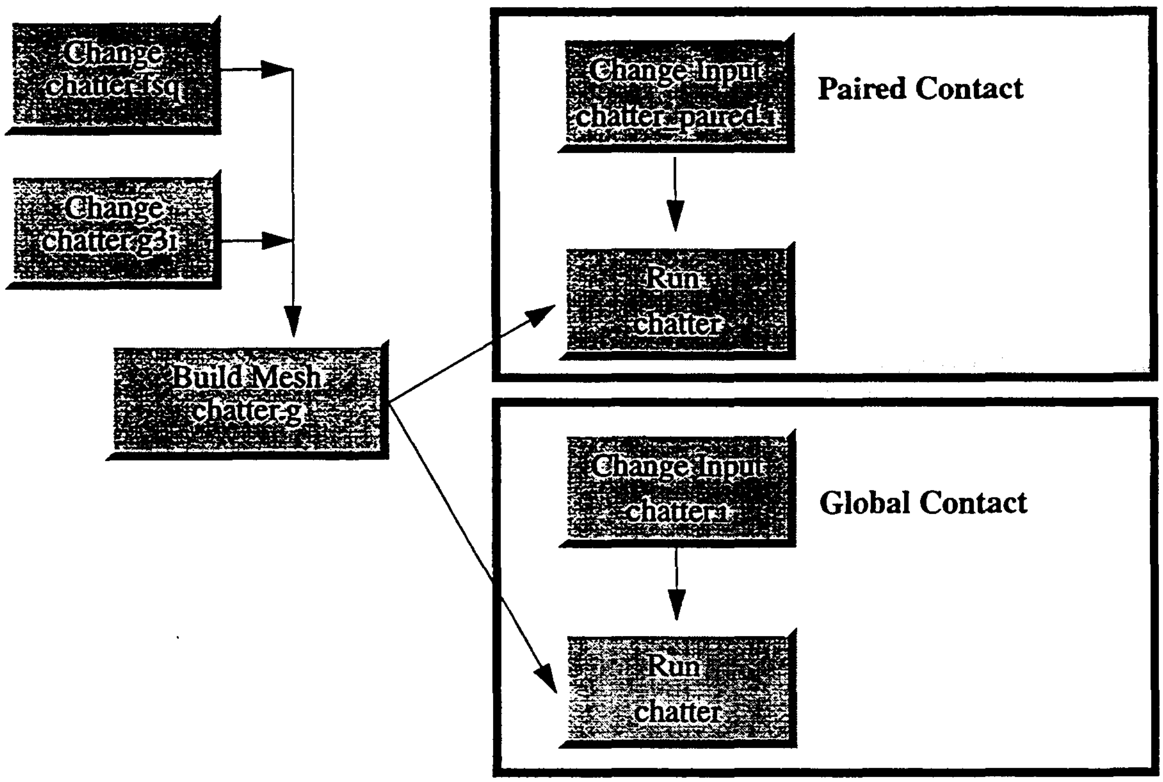


Figure 6 **Example template for building the mesh and running the example.**

Mesh Generation

The mesh was generated using FASTQ and GEN3D. The following files were used:

```
chatter.fsq - FASTQ input file
chatter.g3i - GEN3D input file
```

The mesh can be made using the Makefile with

```
make chatter.g
```

chatter.fsq

```
title
 contact chatter
$ {r=.5} {hx=1} {hy=1}

point 1 0. 0.

point 2 {r} 0.
point 3 {-r} 0.

line 1 str 1 2 0 5
line 2 circ 2 3 1 12
line 3 str 3 1 0 5

region 1 1 -1 -2 -3

point 10 {r} {0}
point 11 {hx} {0}
point 12 {hx} {hy}
point 13 {-hx} {hy}
point 14 {-hx} {0}
point 15 {-r} {0}

line 10 str 10 11 0 5
line 11 str 11 12 0 10
line 12 str 12 13 0 20
line 13 str 13 14 0 10
line 14 str 14 15 0 5
line 15 circ 10 15 1 12

region 2 2 -10 -11 -12 -13 -14 -15

scheme 1 c6s
scheme 2 x


sidebc 1 15
sidebc 2 2

sidebc 100 3 1

linebc 11 12

exit
```

chatter.g3i

```
translate 10 1.
nset front 501
nset back 502
exit
```

Shell Beam

Keywords beam bending, shell elements, pressure load

Description

For this example the structure is an elastic cantilever beam under uniform pressure. Two different loading cases are considered. In the first case the beam is loaded with a pressure of 0.1 psi, yielding small deformations. In the second case the beam is loaded with a pressure of 2.85 psi, causing large rotations. The problem parameters are listed in Table I.

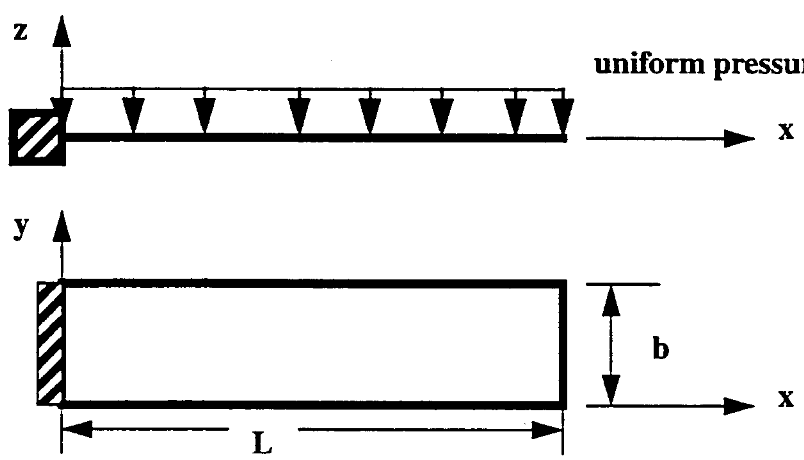


Figure 1 **Schematic of the example model**

Table I **Problem Parameters for Cantilever Beam**

Parameter	Value
Length (L)	10 inch
Width (b)	1.0 inch
Thickness (h)	1.0 inch
Density (rho)	1.024e-6 lb sec^2/in^4
Young's Modulus (E)	1.2e4 psi
Poisson's Ratio	0.2

Finite Element Model

The finite element model used only ten elements, one side set for the pressure load, and one node set for the no displacement boundary condition. A schematic of the mesh is shown in Figure 2.

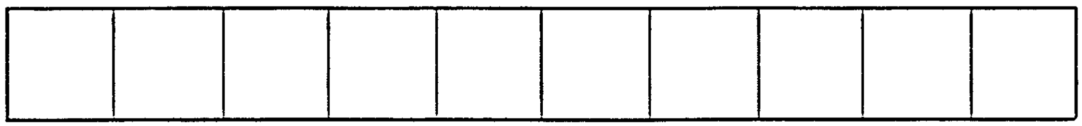


Figure 2 **Finite Element Model Mesh**

Results and Corroborative Data

The maximum end deflections and the natural period are compared with the results from a triangular plate element [Belytschko, T. and Marchertas, A.H., 1974] and a beam element [Belytschko, T., Schwer, L. and Klein, M.J., 1977]. In addition, results from the analytical solution [Timoshenko, S. and Goodier, J.N., 1970] are given for comparison of the small deformation problem. Results indicate that the quadrilateral shell element compares very favorably with the other solutions for both the small and large deformation problems.

Figure 3 shows a plot of the deformation at the end of the beam for the 2.85 psi loading case. The deflected shape of the large deformation cantilever beam is shown in Figure 4. Table II compares the maximum tip deflection of the shell cantilever beam.

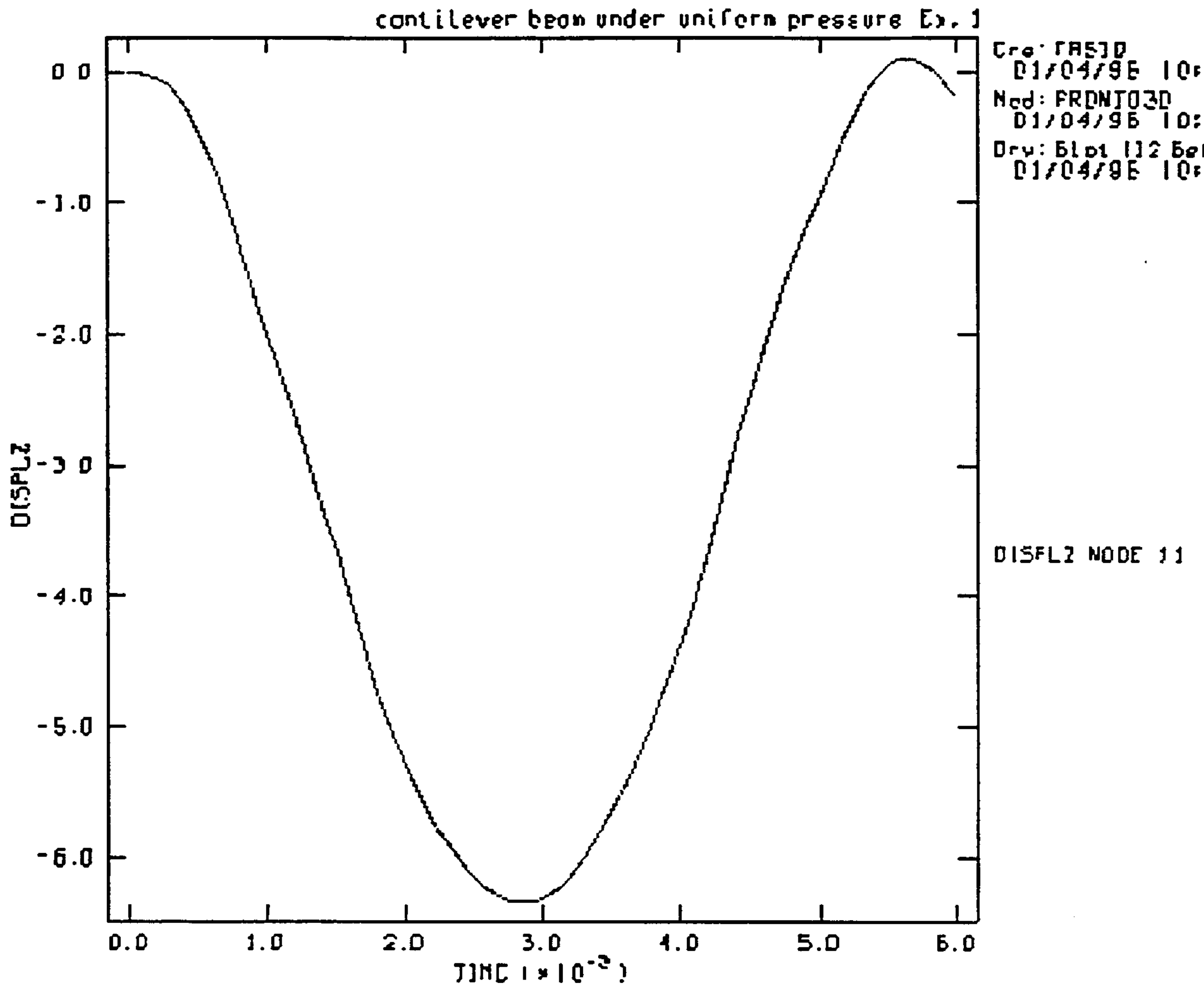


Figure 3 Displacement at tip of beam as a function of time for 2.85 psi loading.

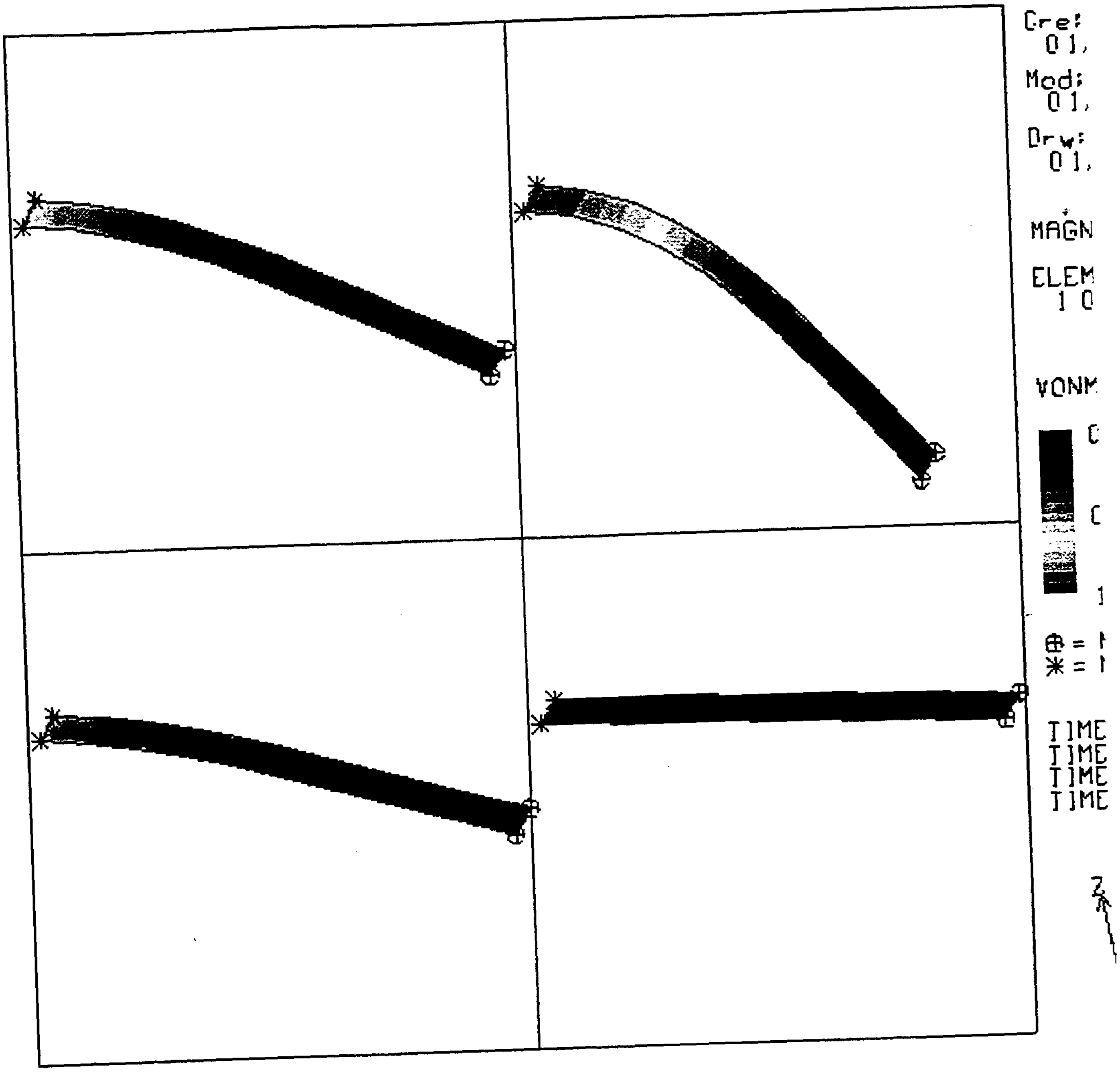


Figure 4 Deformed shape of beam for 2.85 psi loading.

Table II Comparison of Results for Cantilever Beam

	Nodes	Elements	Maximum Tip Deflection (m)	Period (sec)
P = 0.01 psi				
PRONTO shell element	12	10	.0256	5.76e-3
Triangular Shell element	12	20	.0241	5.66e-3

Table II **Comparison of Results for Cantilever Beam**

	Nodes	Elements	Maximum Tip Deflection (m)	Period (sec)
Beam element	6	5	.0253	5.81e-3
Analytic			.0250	5.72e-3
P= 2.85 psi				
PRONTO shell element	12	10	6.35	5.76e-3
Triangular Shell element	12	20	6.08	5.59e-3
Beam element	6	5	6.32	5.58e-3

Finite Element Input Data

shell_beam.i

```
Title
   cantilever beam under uniform pressure
{include(problem.size)}
Termination Time, {tend=6.e-3*sec}
Output Time, {tend/100}
Plot Time, {tend/100}
Plot Nodal, displacement, acceleration, velocity
Plot Element, vonmises
Plot History variable=displ node=11 nam=n11 component=z
$ fixed end boundary condition
No Displacement, x, 4
No Displacement, y, 4
No Displacement, z, 4
No Rotation, x, 4
No Rotation, y, 4

Pressure, 100, 1, {pressure}
Function,1
   {0*sec}, 1
   {1*sec}, 1
End

Shell Hourglass .03,.03,.001
Shell Integration {ninteg} {type}

Material, 1, elastic, {1.024e-6*lbf*sec^2/in^4}
   youngs modulus, {1.2e4*psi}
   poissons ratio, .2
End
Exit
```

The problem.size file for this problem is set up so that the user can change the problem units, the mesh size, pressure loading, and the integration type.

Problem Template

Figure 5 shows an outline of how this problem is constructed using the SEACAS software system. Corresponding text files are included in the Mesh Generation and the Finite Element Input Data sections.

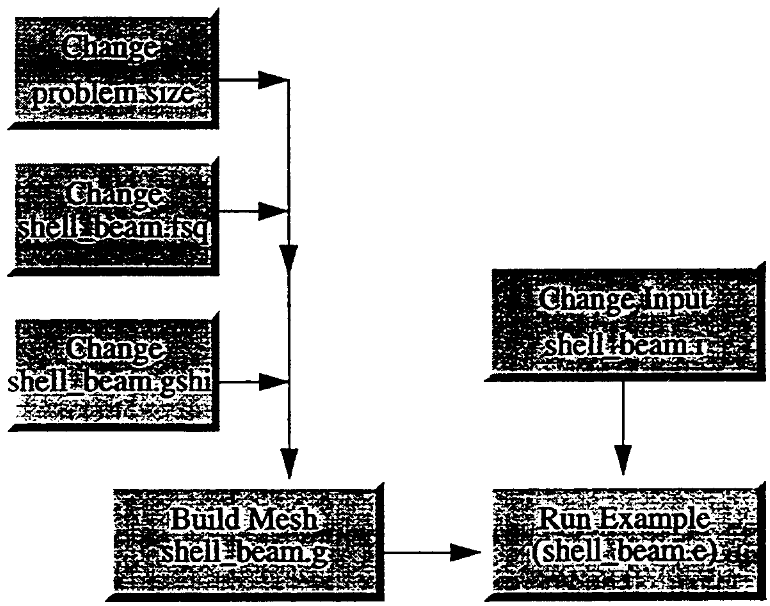


Figure 5 **Example template for building the mesh and running the example.**

Mesh Generation

The mesh was generated using FASTQ and GENSHELL. The following files were used:

```
shell_beam.fsq - FASTQ input file
shell_beam.gshi - GENSHELL input file
```

The mesh can be made using the Makefile with

```
make shell_beam.g
```

The file problem.size contains parameters for the mesh size and is included in each of the mesh files using APREPRO.

<u>problem.size</u>

```
$ try different units: "in-lbf-s" "SI" "shock"
$     {ECHO(OFF)} {Units("in-lbf-s")} {ECHO(ON)}
$
$ mesh size:
$   {intervals_x = 10}
$   {intervals_y = 1}
$
$ applied pressure:
$   {pressure = 2.85*psi}
$
$ try some different rulls : Gauss, Labatto, Trapezoid
$   {ninteg = 5} {type ="Labatto"}
```

shell beam.fsq

```
title
cantilever beam under uniform pressure  Ex. 1
{include(problem.size)}
point    1     0.    0.
point    2    10.    0.
point    3    10.    1.
point    4     0.    1.
poinbc  12     2
poinbc  13     3
line     1     str   1     2     0     {intervals_x}
line     2     str   2     3     0     {intervals_y}
line     3     str   3     4     0     {intervals_x}
line     4     str   4     1     0     {intervals_y}
region   1    1     -1    -2    -3     -4
nodebc   1    1
nodebc   2    2
nodebc   3    3
nodebc   4    4
exit
```

shell beam.gshi

```
translate 1.
ssets front 100
exit
```

Cylindrical Panel

Keywords beam bending, shell elements, pressure load

Description

This example considers an elastic-plastic cylindrical panel, part of which has an initial velocity radially inward. The panel is fixed along the bottom edge and simply supported at the ends. The results from the computation are compared with experimental results from tests preformed at Wright-Patterson AFB [Balmer, H.A. and Witmer, E.A., 1964].

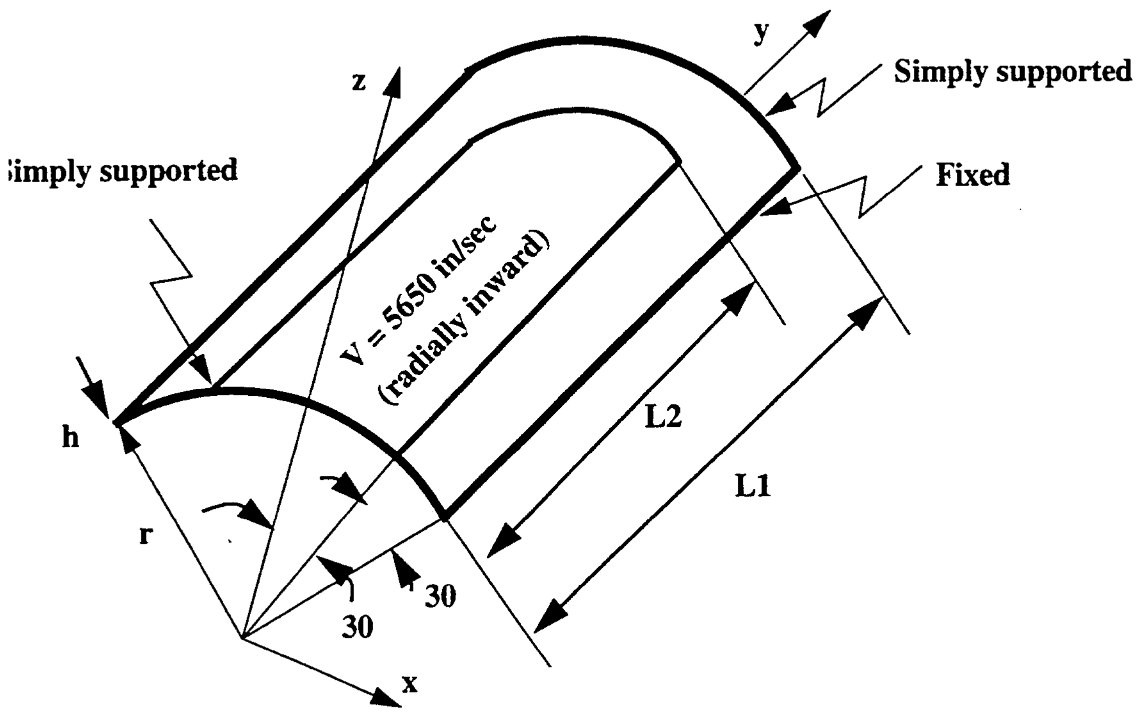


Figure 1 **Schematic of the example model.**

The problem parameters are listed in Table I.

Table I **Problem Parameters for Cylindrical Panel**

Parameter	Value
Length (L1)	12.56 inch
Explosive length (L2)	10.205 inch
Radius (b)	2.938 inch

Table I Problem Parameters for Cylindrical Panel

Parameter	Value
Thickness (h)	0.125 inch
Density (rho)	2.5e-4 lb sec^2/in^4
Young's Modulus (E)	1.05e7 psi
Poisson's Ratio	0.33
Yield Stress	4.4e6 psi
Hardening Modulus	0 psi
Inital Velocity	5650 in/sec

Finite Element Model

Due to symmetry only one-half of the panel was modeled with 2048 shell elements (32 along the circumference and 64 along the length). Three integration points were used through the thickness. The finite element model used three node sets for: 1) the no displacement boundary condition, 2) the symmetry boundary condition, 3) the simple support boundary condition. The initial velocity boundary condition required a radially inward normal that was generated by defining different node sets along the length of the explosive. The corresponding velocity was then matched with the Initial Velocity Nodeset command in the PRONTO input. A schematic of the mesh is shown in Figure 2.

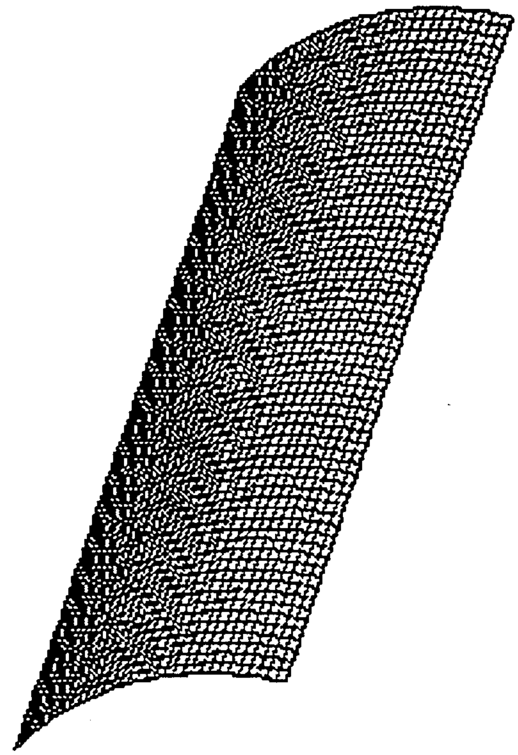


Figure 2 **Finite Element Model Mesh**

Results and Corroborative Data

Figure 3 shows various stages of deformation for the panel. A plastic hinge forms along the edge of the impulse loading region.

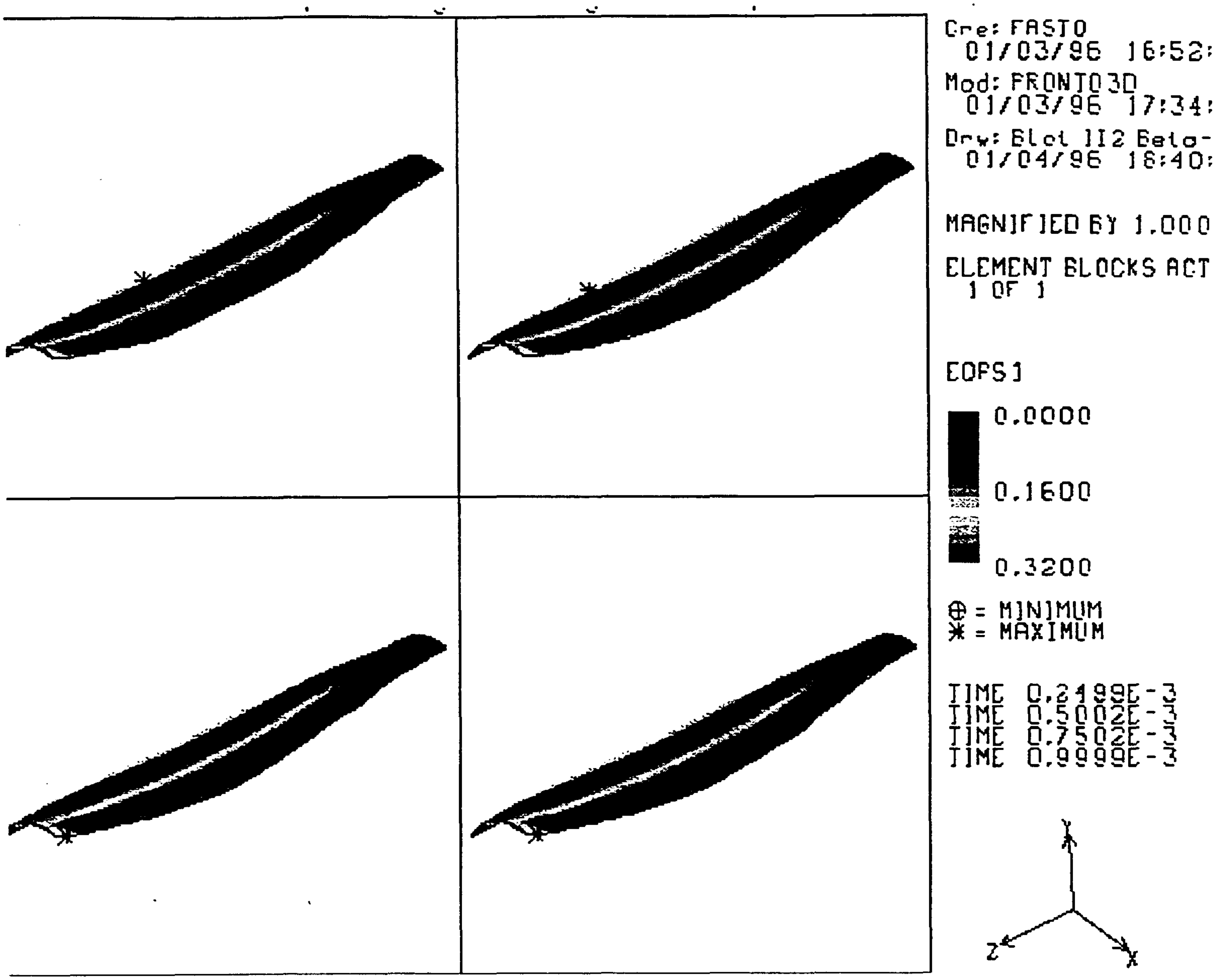


Figure 3 **Plot of equivilent plastic strain on deformed shape of cylindrical panel.**

In Figure 4 the time histories for vertical deflection at two points on the crown line are plotted. These results compared well with experimental results from [Balmer, H.A. and Witmer, E.A., 1964]. The results for the shell element have improved from what was published in [Bergmann, V.L., 1991]. While the results in [Bergmann, V.L., 1991] were in good agreement at $y = 6.28$ inch; there was some disagreement at $y = 9.42$ inch. The present results shown in Figure 4 are in much better agreement with the experiment. The reason for error was a miscalculated rotation in the original formulation that has since been corrected. The final deflected shape of the crown line (plotted in Figure 5 relative to the undeformed shape) agrees well with the experimental results.

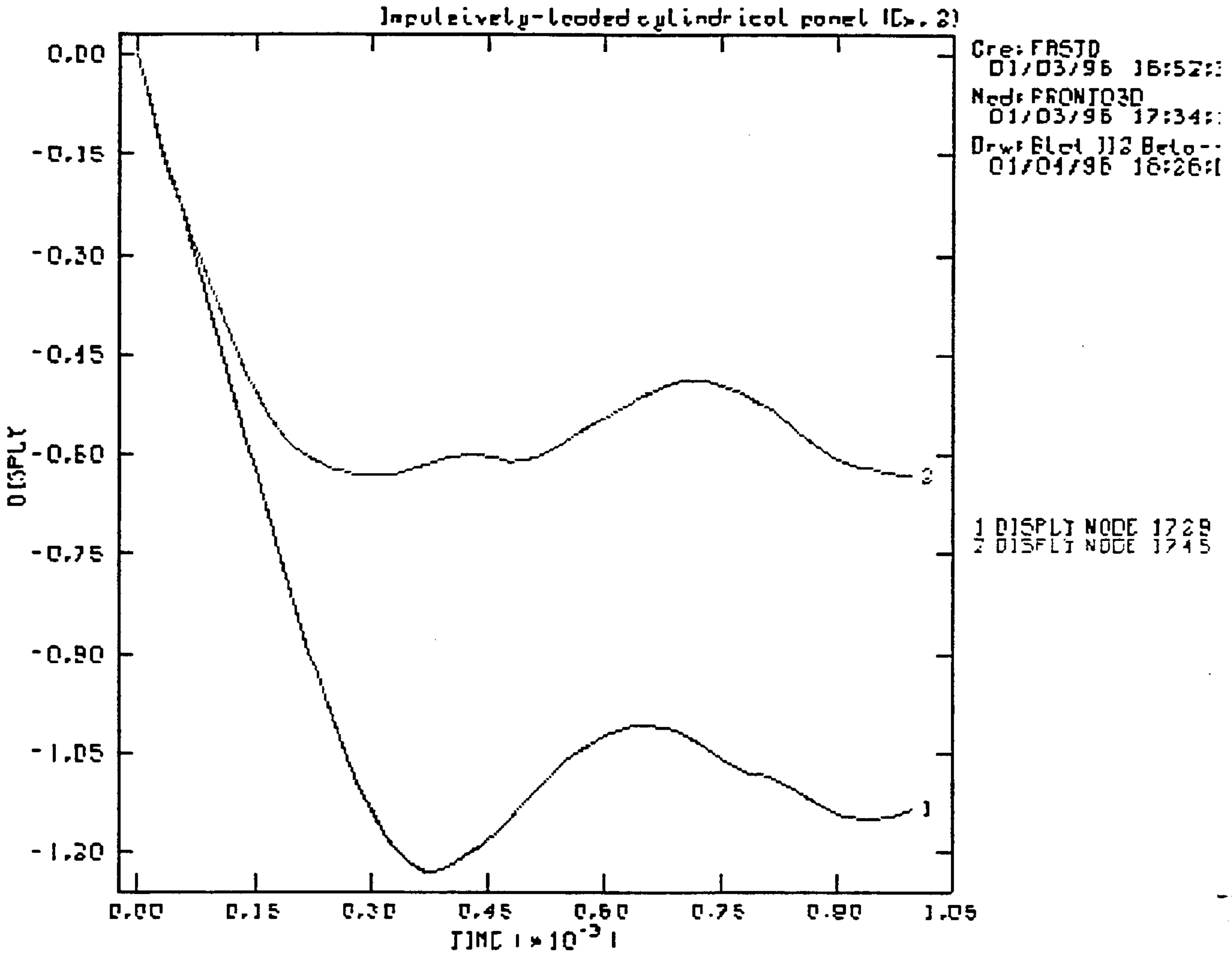


Figure 4 **Time history of displacement at two points on the crown line.**

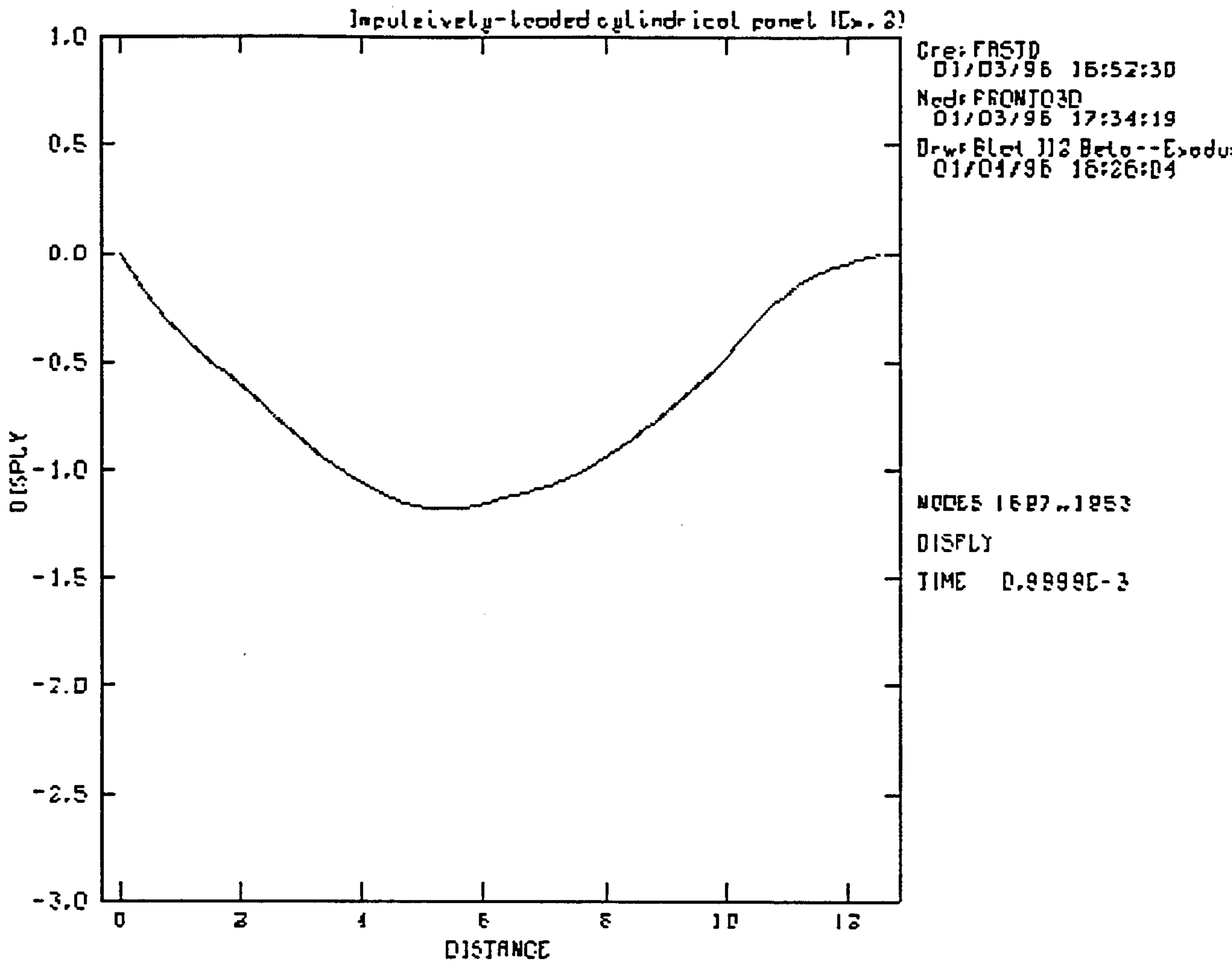


Figure 5 **Final Deformed Shape of Crown Line (t = 1.0e-3 sec).**

The advantages of shell elements over brick elements for this problem are apparent from Table II. Three brick elements were used through the thickness, 32 elements along the circumference, and 64 elements along the length, for a total of 6,144 hexahedral elements. The brick and shell element meshes have the same number of integration points where the constitutive relations are computed. Table II shows that the shell elements not only yield better results than the hex elements, but also require significantly less run time and memory space.

Table II Comparison of Resource Requirements and Accuracy for Cylindrical Panel

	Shells	Shells	Bricks	Experimental
Nodes	2145		8580	
Elements	2048		6144	

Table II **Comparison of Resource Requirements and Accuracy for Cylindrical Panel**

	Shells	Shells	Bricks	Experimental
Maximum Deflection y= 6.28 in	-1.20		-0.98	-1.25
Maximum Deflection y= 9.42 in	-0.82		-0.47	-0.72
Machine	CRAY YMP	CJ90	CRAY YMP	
CPU Time (sec)	136	213	907	
Memory Space (words)	457,000	2,488,320	78,320	

Finite Element Input Data

<u>shell cyl panel.i</u>

```
Title
   Impulsively-loaded cylindrical panel (Ex. 2)
No Displacement x 301
No Displacement y 302
No Displacement z 303
No Rotation x  304
No Rotation y  305
No Rotation z  306
Initial Velocity Nodeset  102  2825.000   -4893.044  0.
Initial Velocity Nodeset  103  2663.392   -4982.855  0.
Initial Velocity Nodeset  104  2498.931   -5067.331  0.
Initial Velocity Nodeset  105  2331.795   -5146.381  0.
Initial Velocity Nodeset  106  2162.161   -5219.919  0.
Initial Velocity Nodeset  107  1990.213   -5287.868  0.
Initial Velocity Nodeset  108  1816.133   -5350.155  0.
Initial Velocity Nodeset  109  1640.108   -5406.713  0.
Initial Velocity Nodeset  110  1462.328   -5457.481  0.
Initial Velocity Nodeset  111  1282.981   -5502.405  0.
Initial Velocity Nodeset  112  1102.260   -5541.437  0.
Initial Velocity Nodeset  113   920.359   -5574.535  0.
Initial Velocity Nodeset  114   737.473   -5601.663  0.
Initial Velocity Nodeset  115   553.797   -5622.794  0.
Initial Velocity Nodeset  116   369.528   -5637.903  0.
Initial Velocity Nodeset  117   184.863   -5646.975  0.
Initial Velocity Nodeset  118     0.000   -5650.000  0.
Function,1
   0  1
   1  1
End
Shell Hourglass 0., 0., 0.
Termination Time, 1.e-3
Output Time 1.e-5
Plot Time, 1.e-5
Material, 1, elastic plastic, 2.5e-4
   youngs modulus = 1.05e7
   poissons ratio = 0.33, yield stress = 44000.
```

```
   hardening modulus = 0., beta = 1.
End
Plot History variable = displ node = 1729 component = y name = n1729
Plot History variable = displ node = 1745 component = y name = n1745
Plot Element = vonmises
Plot Nodal = displacement
Plot State = eqps
Exit
```

Problem Template

Figure 6 shows an outline of how this problem is constructed using the SEACAS software system. Corresponding text files are included in the Mesh Generation and the Finite Element Input Data sections.

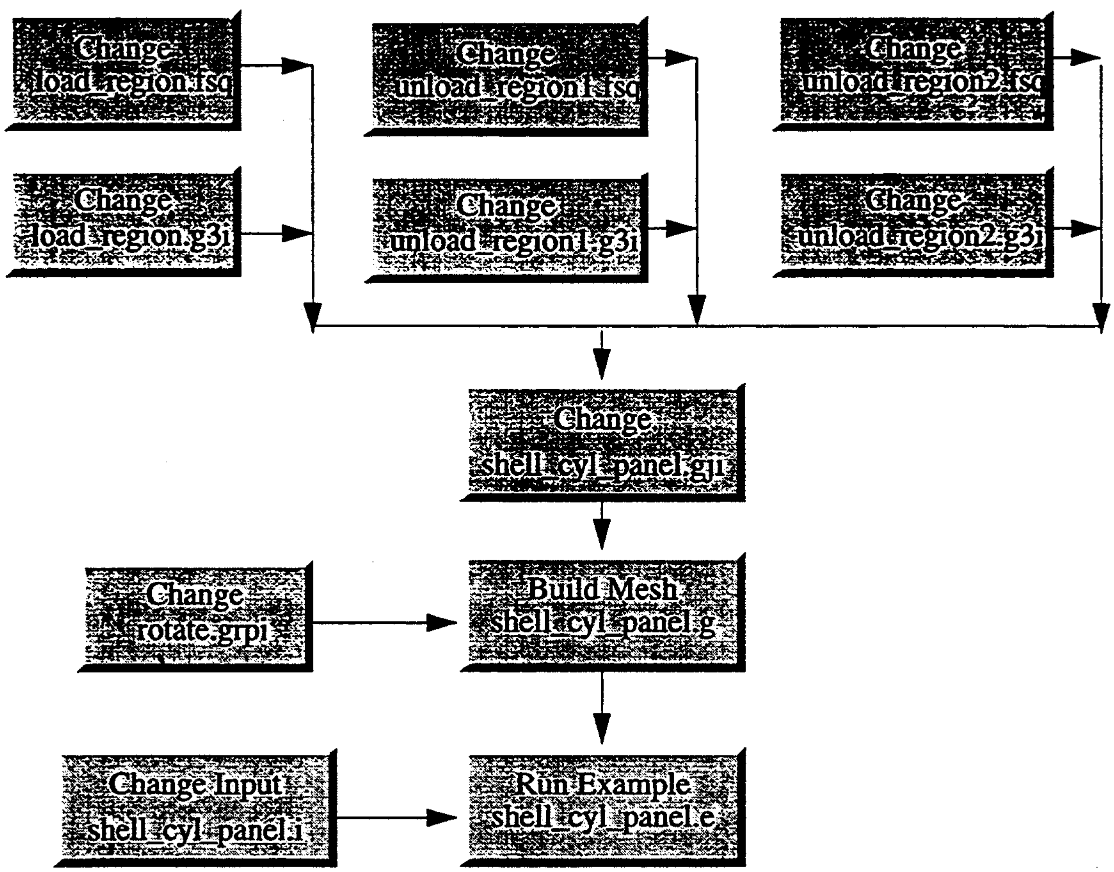


Figure 6 **Example template for building the mesh and running the example.**

Mesh Generation

The mesh was generated using FASTQ, GEN3D, GJOIN, and GREPOS. The following files were used:

```
unload_region1.fsq, unload_region2.fsq, and load_region.fsq - FASTQ input
        files
unload_region1.g3i, unload_region2.g3i, and load_region.g3i - GEN3D input
        files
```

```
shell_cyl_panel.gji - GJOIN input file
rotate.grpi - GREPOS input file
```

The mesh can be made using the Makefile with

```
make shell_cyl_panel.g
```

unload region1.fsq

```
title
Cylindrical panel with impulse loading  Ex. 2
point 1  0.       2.938
point 2  1.46900  2.54438
point 3  1.55146  2.49496
point 4  1.63227  2.44286
point 5  1.71132  2.38814
point 6  1.78854  2.33087
point 7  1.86385  2.27110
point 8  1.93716  2.20891
point 9  2.00839  2.14434
point 10 2.07748  2.07748
point 11 2.14434  2.00839
point 12 2.20891  1.93716
point 13 2.27110  1.86385
point 14 2.33087  1.78854
point 15 2.38814  1.71132
point 16 2.44286  1.63227
point 17 2.49496  1.55146
point 18 2.54438  1.46900
point 100    0.     0.
line 1 circ  1    2   -100   16
poinbc  1   1
poinbc 301  1
poinbc 302  1
poinbc 303  1
poinbc 304  1
poinbc 305  1
poinbc 306  1
barset  1   1   0    1
exit
```

unload region2.fsq

```
title
Cylindrical panel with impulse loading  Ex. 2
point 1  0.       2.938
point 2  1.46900  2.54438
point 3  1.55146  2.49496
point 4  1.63227  2.44286
point 5  1.71132  2.38814
point 6  1.78854  2.33087
point 7  1.86385  2.27110
point 8  1.93716  2.20891
point 9  2.00839  2.14434
point 10 2.07748  2.07748
point 11 2.14434  2.00839
point 12 2.20891  1.93716
point 13 2.27110  1.86385
point 14 2.33087  1.78854
point 15 2.38814  1.71132
point 16 2.44286  1.63227
point 17 2.49496  1.55146
point 18 2.54438  1.46900
point 100    0.     0.
```

```
poinbc  1     1
poinbc 18     18
poinbc 301    1   18
poinbc 302    1
poinbc 303    1
poinbc 304    1
poinbc 305    1   18
poinbc 306    1   18
line  1 circ  2    3   -100    1
line  2 circ  3    4   -100    1
line  3 circ  4    5   -100    1
line  4 circ  5    6   -100    1
line  5 circ  6    7   -100    1
line  6 circ  7    8   -100    1
line  7 circ  8    9   -100    1
line  8 circ  9   10   -100    1
line  9 circ 10   11   -100    1
line 10 circ 11   12   -100    1
line 11 circ 12   13   -100    1
line 12 circ 13   14   -100    1
line 13 circ 14   15   -100    1
line 14 circ 15   16   -100    1
line 15 circ 16   17   -100    1
line 16 circ 17   18   -100    1
line 17 circ  1    2   -100    16
barset  1   1   0    1
barset  2   1   0    2
barset  3   1   0    3
barset  4   1   0    4
barset  5   1   0    5
barset  6   1   0    6
barset  7   1   0    7
barset  8   1   0    8
barset  9   1   0    9
barset  10   1   0    10
barset  11   1   0    11
barset  12   1   0    12
barset  13   1   0    13
barset  14   1   0    14
barset  15   1   0    15
barset  16   1   0    16
barset  17   1   0    17
exit
```

load_region.fsq

```
title
Cylindrical panel with impulse loading  Ex. 2
point 1  0.       2.938
point 2  1.46900  2.54438
point 3  1.55146  2.49496
point 4  1.63227  2.44286
point 5  1.71132  2.38814
point 6  1.78854  2.33087
point 7  1.86385  2.27110
point 8  1.93716  2.20891
point 9  2.00839  2.14434
point 10 2.07748  2.07748
point 11 2.14434  2.00839
point 12 2.20891  1.93716
point 13 2.27110  1.86385
point 14 2.33087  1.78854
point 15 2.38814  1.71132
```

```
point 16 2.44286  1.63227
point 17 2.49496  1.55146
point 18 2.54438  1.46900
point 100    0.     0.
poinbc 18   18
poinbc  102   2
poinbc  103   3
poinbc  104   4
poinbc  105   5
poinbc  106   6
poinbc  107   7
poinbc  108   8
poinbc  109   9
poinbc  110  10
poinbc  111  11
poinbc  112  12
poinbc  113  13
poinbc  114  14
poinbc  115  15
poinbc  116  16
poinbc  117  17
poinbc  118  18
poinbc  301  18
poinbc  305  18
poinbc  306  18
line  1 circ  2     3    -100    1
line  2 circ  3     4    -100    1
line  3 circ  4     5    -100    1
line  4 circ  5     6    -100    1
line  5 circ  6     7    -100    1
line  6 circ  7     8    -100    1
line  7 circ  8     9    -100    1
line  8 circ  9    10    -100    1
line  9 circ 10    11    -100    1
line 10 circ 11    12    -100    1
line 11 circ 12    13    -100    1
line 12 circ 13    14    -100    1
line 13 circ 14    15    -100    1
line 14 circ 15    16    -100    1
line 15 circ 16    17    -100    1
line 16 circ 17    18    -100    1
barset  1   1   0    1
barset  2   1   0    2
barset  3   1   0    3
barset  4   1   0    4
barset  5   1   0    5
barset  6   1   0    6
barset  7   1   0    7
barset  8   1   0    8
barset  9   1   0    9
barset  10   1   0    10
barset  11   1   0    11
barset  12   1   0    12
barset  13   1   0    13
barset  14   1   0    14
barset  15   1   0    15
barset  16   1   0    16
exit
```

unload_region1.g3i

```
translate 52 10.205
nsets front 401 402  403
```

```
attribute 1 .125
exit
```

unload_region2.g3i

```
translate 12 2.355
nsets back 401 402 403
offset 0. 0. -10.205
attribute 1 0.125
exit
```

load_region.g3i

```
translate 52 10.205
nsets front 401 402 403
attribute 1 .125
exit
```

shell_cyl_panel.gji

```
unload_region1.g
load_region.g
y
n
1.e-5
n
add
unload_region2.g
y
n
1.e-5
n
nsets
combine 301 401
combine 302 402
combine 303 403
combine 304 404
combine 305 405
exit
finish
temp.g
exit
```

rotate.grpi

```
revolve z 60
exit
```

Shell Tearing

Keywords tearing, element deletion, shell elements, pull test

Description

An increasing number of structural analyses are addressing the question of material failure. PRONTO 3D allows elements to fail (adaptive element deletion or death) during the analysis when a specified criterion is reached. When an element fails, the element is assumed to be incapable of sustaining stress. For an element that fails, all components of stress within the element are reduced to zero over a set number of time steps (user-supplied, default = 5). This reduction in stress over more than one time step allows for dissipation and redistribution of internal elastic strain energy to adjacent elements as kinetic energy.

This example considers a notched tension test modeled using shell elements. The elements are deleted based on a criterion defined in the PLH STRENGTH model [Stone, C. M. and Wellman, G. W., 1993], [Wellman, G. W., 1993]. For shell elements the death criterion is based on a sum over all the integration points. Therefore, the failure criterion must be met at all integration points before the element is removed from the problem. The problem parameters are listed in Table I.

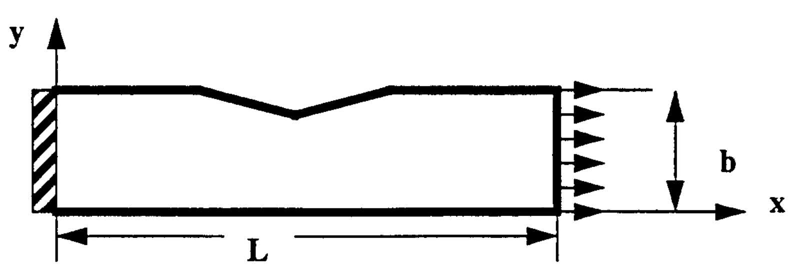


Figure 1 **Schematic of the example model**

Table I **Problem Parameters for Shell Tearing**

Parameter	Value
Length (L)	10 inch
Width (b)	1.0 inch
Thickness (h)	1.0 inch
Density (rho)	1.024e-6 lb sec^2/in^4

Table I Problem Parameters for Shell Tearing

Parameter	Value
Young's Modulus (E)	1.2e4 psi
Poisson's Ratio	0.2

Finite Element Model

The finite element model is shown in Figure 2. The mesh was generated using the paving algorithm in FASTQ. This algorithm allows more elements to be placed in the region of expected failure, which will minimize the errors associated with the element deletion. In real structures the failure zone will be very small compared to the element size in typical finite element analysis. The results are expected to be mesh dependent.

The PLH strength model defines a state variable "TDECAY1" that represents the sum of the damage terms on all integration layers of the shell. Here the elements were deleted from the analysis after this term reaches a value of 0.15. The element variable STATUS is output to the EXODUS database so that the visualization software can tell which elements are active and which are inactive.

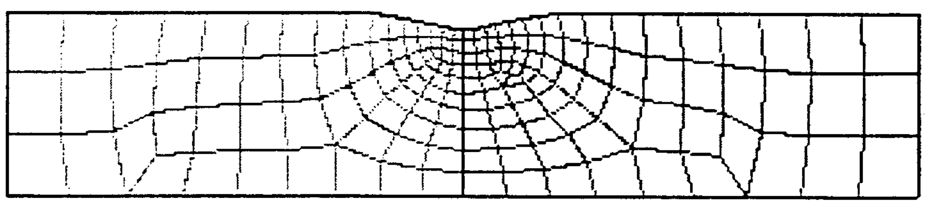


Figure 2 **Finite Element Model Mesh**

Results and Corroborative Data

Figure 3 shows a plot of the deformed shape of the element as the specimen is pulled. The deleted elements are not included in the plot. At the time shown in the plot, the "deleted zone" has progressed about halfway through the thickness of the specimen. Some necking in the region of the notch developed before the elements started to fail.

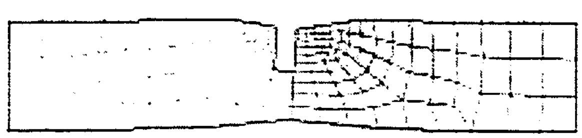


Figure 3 **Deformed shape of beam.**

Elements can be deleted based on element variables, such as equivalent plastic strain, maximum principle stress, or element volume. In the PLH STRENTH model and in the SANDIA DAMAGE model, a damage parameter is used to reduce the material strength in a controlled way. Once the element strength has been reduced, deleting elements simply removes the element from the calculation and prevents excessive distortions from causing numerical problems. As the material strength is reduced, the material is said to be undergoing strain softening. When this softening behavior occurs, the central difference integrator becomes unconditionally unstable. Some numerical viscosity can help to maintain stability by absorbing some of the internal energy of the softening element.

If elements are deleted using ad hoc methods, such as deleting the element based on a maximum principle strain, then the element is in effect softening. The softening behavior can be adjusted by changing the number of time steps over which an element is deleted or the amount of viscous energy absorbed by the deleted element.

Finite Element Input Data

shell_beam.i

```
Title
   tension tearing test  small mesh
Termination Time, 4.5e-3
Output Time, 1.e-5
Plot Time, 1.e-4
Plot Nodal, displacement, acceleration, velocity
Plot Element, vonmises
Plot State, eqps1, eqps5, tearing1, tearing5, tdecay1

No Displacement, x, 4
No Displacement, y, 4
No Displacement, z, 4
No Rotation, x, 4
No Rotation, y, 4

Prescribed Velocity x 2 1 1.0
Prescribed Velocity y 2 1 0.
Prescribed Velocity z 2 1 0.

Death, 1, tdecay1, min,.15
Death, 2, tdecay1, min,.15

Function ,1 $ velocity
   0,0.
   1,100000.
End

Material, 1, PLH STRENGTH, 0.000254  $6061-T6 AL
   YOUNGS MODULUS,        9.9E06
   POISSONS RATIO,        0.33
   YIELD STRESS,          42000.
   HARDENING CONSTANT,    29964
   HARDENING EXPONENT,    0.3406
   LUDERS STRAIN,         0.
   FAILURE VALUE,         0.5
   DECAY CONSTANT,        0.9
END
Material, 2, PLH STRENGTH, 0.000254  $6061-T6 AL
   YOUNGS MODULUS,        9.9E06
   POISSONS RATIO,        0.33
   YIELD STRESS,          42000.
   HARDENING CONSTANT,    29964
```

```
    HARDENING EXPONENT,    0.3406
    LUDERS STRAIN,         0.
    FAILURE VALUE,         0.5
    DECAY CONSTANT,        0.9
END

Exit
```

Problem Template

Figure 4 shows an outline of how this problem is constructed using the SEACAS software system. Corresponding text files are included in the Mesh Generation and the Finite Element Input Data sections.

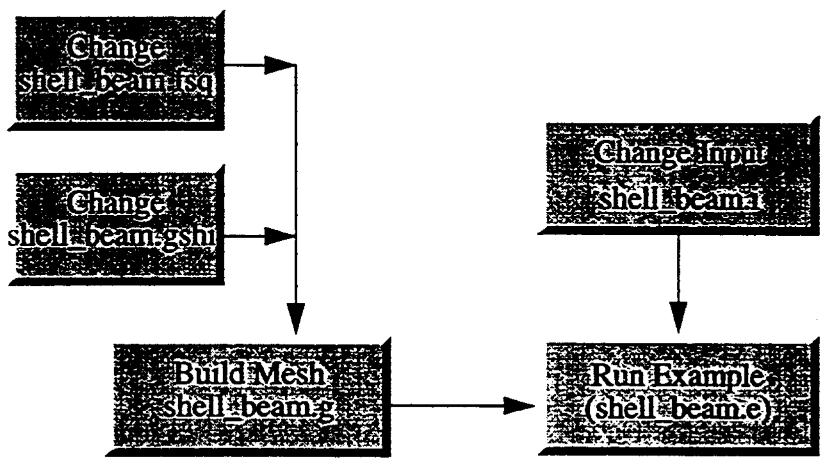


Figure 4 **Example template for building the mesh and running the example.**

Mesh Generation

The mesh was generated using FASTQ and GENSHELL. The following files were used:

```
shell_beam.fsq - FASTQ input file
shell_beam.gshi - GENSHELL input file
```

The mesh can be made using the Makefile with

```
make shell_beam.g
```

shell_beam.fsq

```
title
 tension pull test
point     1      0.      0.
point     2     10.      0.
point     3     10.      2.
point    31      6.      2.
point    32      5.      1.8
point    33      4.      2.
point     4      0.      2.
point    34      5.      0.
poinbc   12        2
poinbc   13        3
$ {size = 8}
line     1    str      1      34      0    {size*9} 0.95
```

```
line     35    str      34        2       0     {size*9} 1.05
line      2    str       2        3       0     {size*3}
line      3    str       3       31       0     {size*9} 0.9
line     31    str      31       32       0     {size*7} 0.9
line     32    str      32       33       0     {size*7} 1.1
line     33    str      33        4       0     {size*9} 1.1
line      4    str       4        1       0     {size*3}
line     34    str      32       34       0     {size*10} 1.1
region   1     1    -2   -3 -31 -34 -35
region   2     2   -34 -32 -33   -4  -1
scheme   1 x6s
scheme   2 x6s
nodebc    1     1
nodebc    2     2
nodebc    3     3
nodebc    4     4
exit
```

shell_beam.gshi

```
translate 1.
ssets front 100
exit
```

Cavity Expansion: Aluminum

Keywords cavity expansion, pressure load, penetration

From: [Warren, T.L. and Tabbara, M.R., 1997]

Description

In these examples we consider the penetration of 6061-T651 aluminum targets by solid spherical-nose, C-300 maraging steel rods launched at striking velocities between 350 and 1200 m/s. These rods have density $\rho_p = 8000$ kg/m^3, shank length $L = 71.12$ mm, nose radius $a = 3.55$ mm, and nominal mass $m = 0.0235$ kg. The target is modeled as a compressible, strain hardening, and strain-rate sensitive material for which the undeformed density, quasistatic yield strength, and dimensionless fitting coefficients required for use with the cavity expansion model are given by [Warren, T.L. and Forrestal, M.J., 1997] as $\rho_0 = 2710$ kg/m^3, $Y = 276$ MPa, $A = 5.0394$, $B = 0.9830$, and $C = 0.9402$, respectively. The finite element mesh used in the following examples is shown in and is comprised of 3,172 nodes and 2,784 elements.

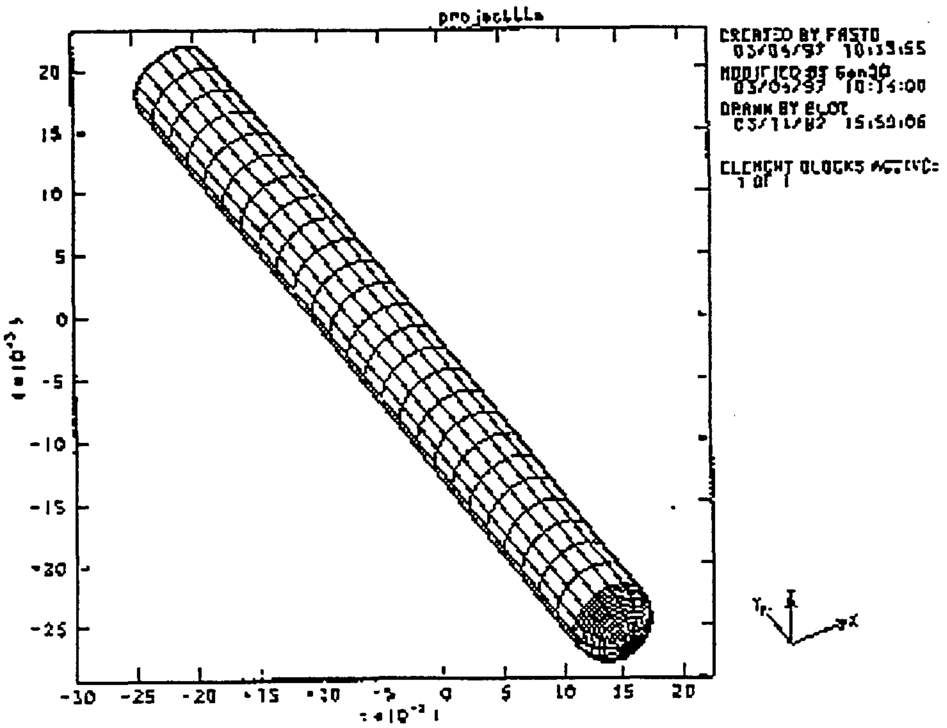


Figure 1 **Finite element mesh of the spherical-nose rod.**

Results and Corroborative Data

We first compare results obtained by PRONTO3D with the experimentally verified analytical model given by [Warren, T.L. and Forrestal, M.J., 1997] to validate the cavity expansion forcing function. In the analytical model the projectile is assumed to be rigid; therefore, we use the RIGID material model in PRONTO3D. An example input file for a striking velocity of 960 m/s is shown in ca_al_rigid.i. The Cavity Expansion command line includes a *side set id* of 100, which represents the surfaces of the nose and shank (excluding the rear of the projectile). A bounding coordinate, b1 = 0, is used to define the free surface, and b2 = -10 m to reflect an unbounded medium. shows the depth of penetration at several striking velocities that are in good agreement with the analytical results. A termination time of 350 ms was used for each of the striking velocities, requiring approximately 22 cpu seconds on a CRAY J-90.

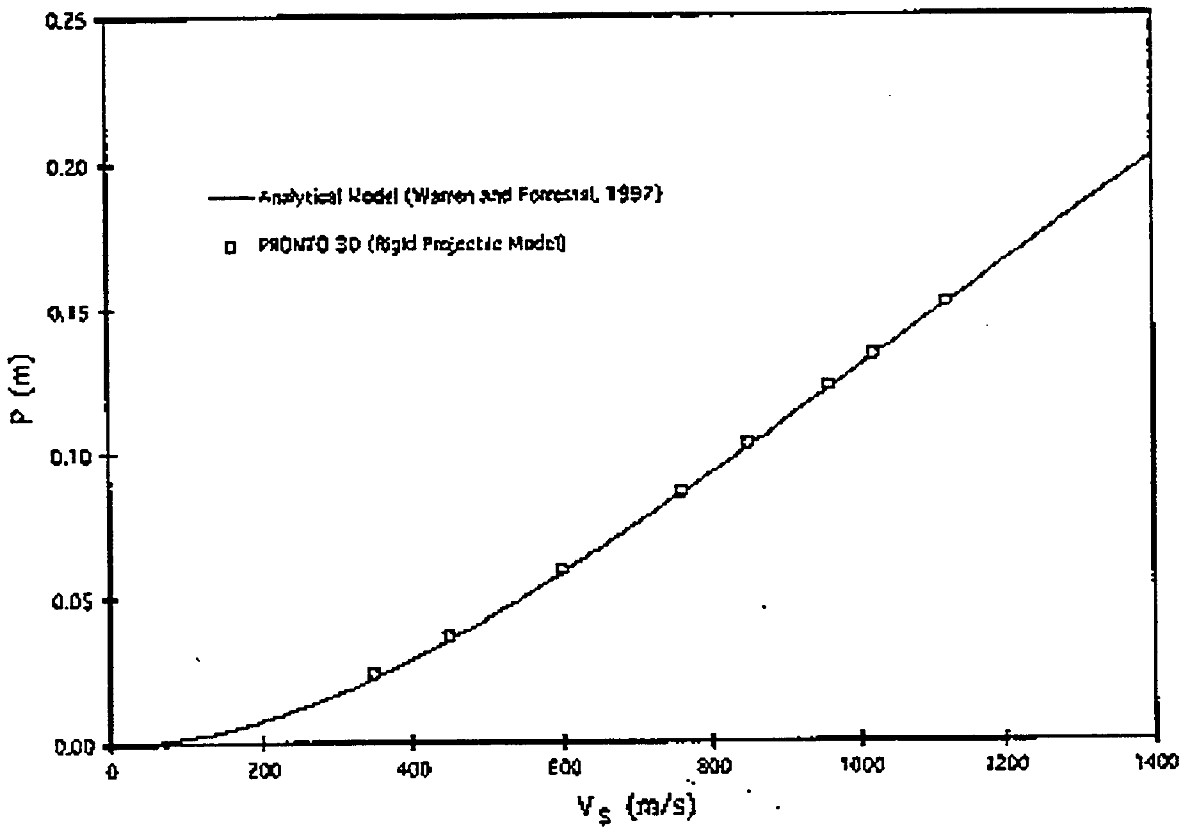


Figure 2 **Depth of penetration versus striking velocity for a rigid projectile.**

Next, we assume the C-300 maraging steel projectiles to behave as an elastic, linear-hardening plastic material and use the ELASTIC PLASTIC material model in PRONTO3D. The Young's modulus, yield strength, and Poisson's ratio for maraging steel are given by [International Nickel Company, Inc., The, 1964] as $E = 189$ GPa, $Y = 2.067$ GPa, and $\nu = 0.3$, respectively. Experimental results obtained by [Chait, R., 1972] from compression tests of C-300 maraging steel at room temperature and a strain rate of $3 \times 10^{-4}\ s^{-1}$ give a strain hardening modulus of $E' = 177.7$ MPa. An example input file using these parameters is shown in ca_al.i for a striking velocity of 960 m/s. Depth of penetration results are compared in with the rigid projectile analytical solution of [Warren, T.L. and Forrestal, M.J., 1997], and also with experimental data obtained by [Forrestal, M.J., Okajima, K., and Luk, V.K., 1988] for C-300 maraging steel and by [Forrestal, M.J., Brar, N.S., and Luk, V.K., 1991] for T-200 maraging steel, which is slightly softer than the C-300 maraging steel. Good agreement is observed between the analytical solution and the PRONTO3D solution for striking velocities below 700 m/s; however, for higher velocities the projectile deformation reduces the depth of penetration. It should be noted that the depth of penetration would be greater if strain-rate effects had been included in the constitutive model for the penetrator. A sequence of deformed penetrator configurations is shown in for a striking velocity of $V_s = 1120$ m/s. It is observed that the penetrator bulges slightly early in the penetration event requiring it to open a larger cavity which reduces the depth of penetration. A termination time of 350 ms was used for each of the striking velocities, requiring approximately 1,482 cpu seconds on a CRAY J-90.

As another example, we consider the oblique impact of a C-300 maraging steel projectile with a 6061-T651 aluminum target. The input file for a 30 degree oblique impact with a zero angle of attack and a striking velocity of 960 m/s is shown in ca_al_30.i. A sequence of deformed penetrator configurations for this striking velocity and angle of obliqueness is shown in . It is observed that the projectile initially bulges slightly and starts to bend due to the nonsymmetric loading. The projectile continues to bend and rotate throughout the penetration event until it finally comes to rest. As in the previous example, a termination time of 350 ms was used, requiring approximately 1,482 cpu seconds on a CRAY J-90. While we have good agreement between data and predictions for normal impacts, at this time we have no oblique impact data for aluminum targets.

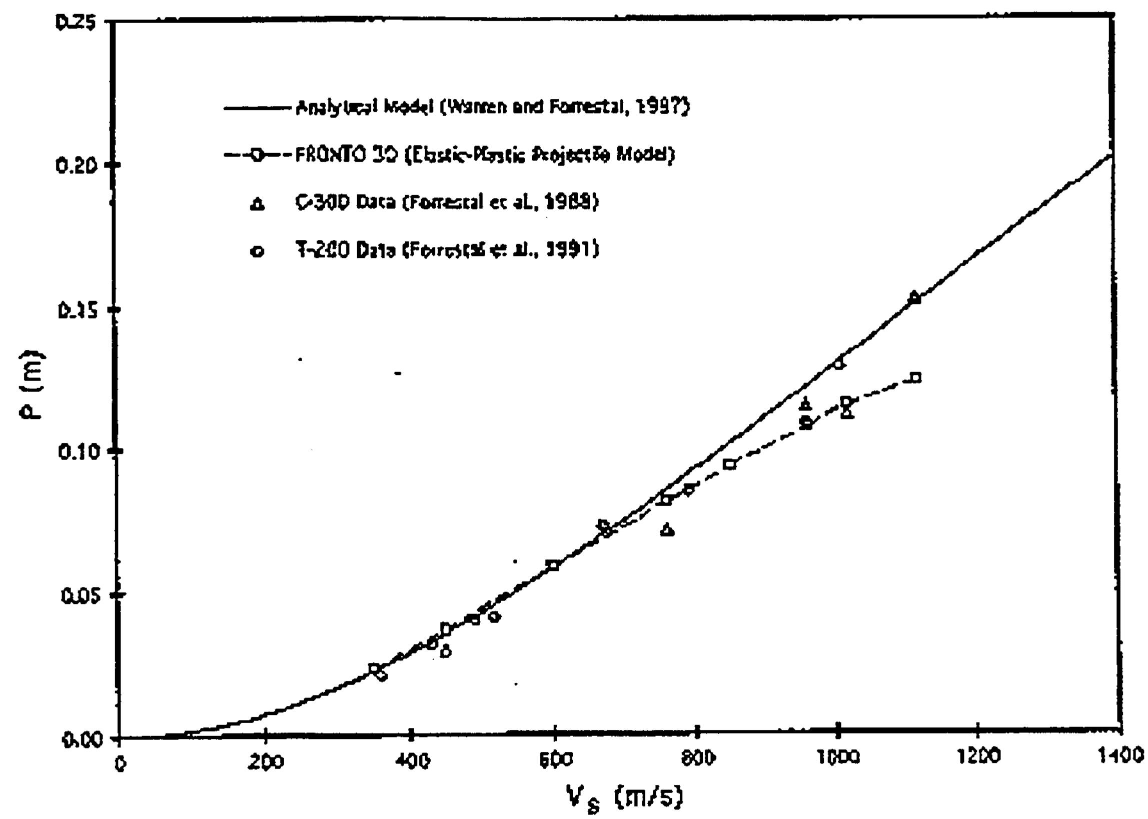


Figure 3 Depth of penetration versus striking velocity for an elastic-plastic projectile.

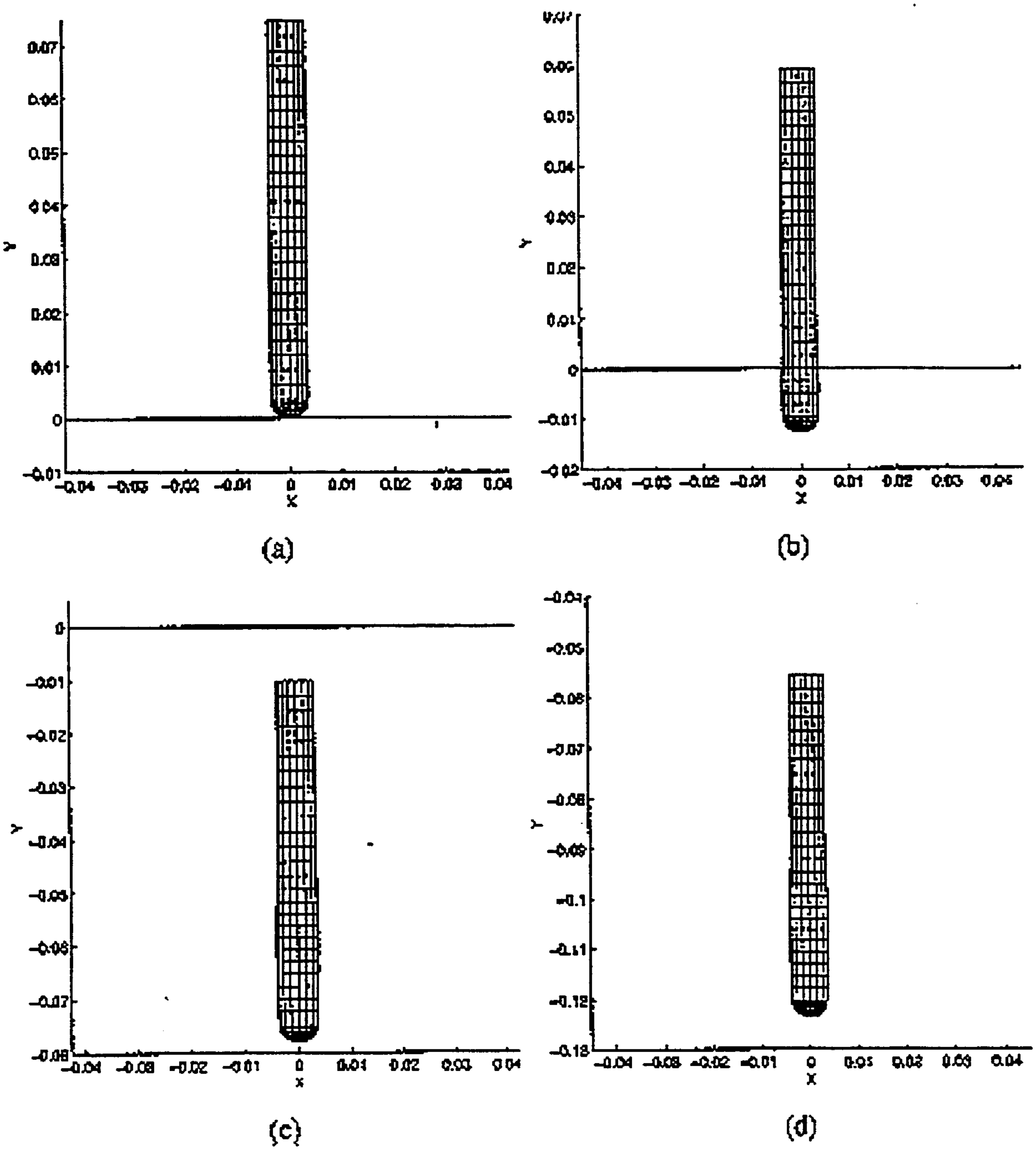


Figure 4 **Projectile deformation for a striking velocity** v_s = 1120 m/s **at: (a) 0.0** μ **s; (b) 14** μ **s; (c) 98** μ **s; and (d) 280** μ **s.**

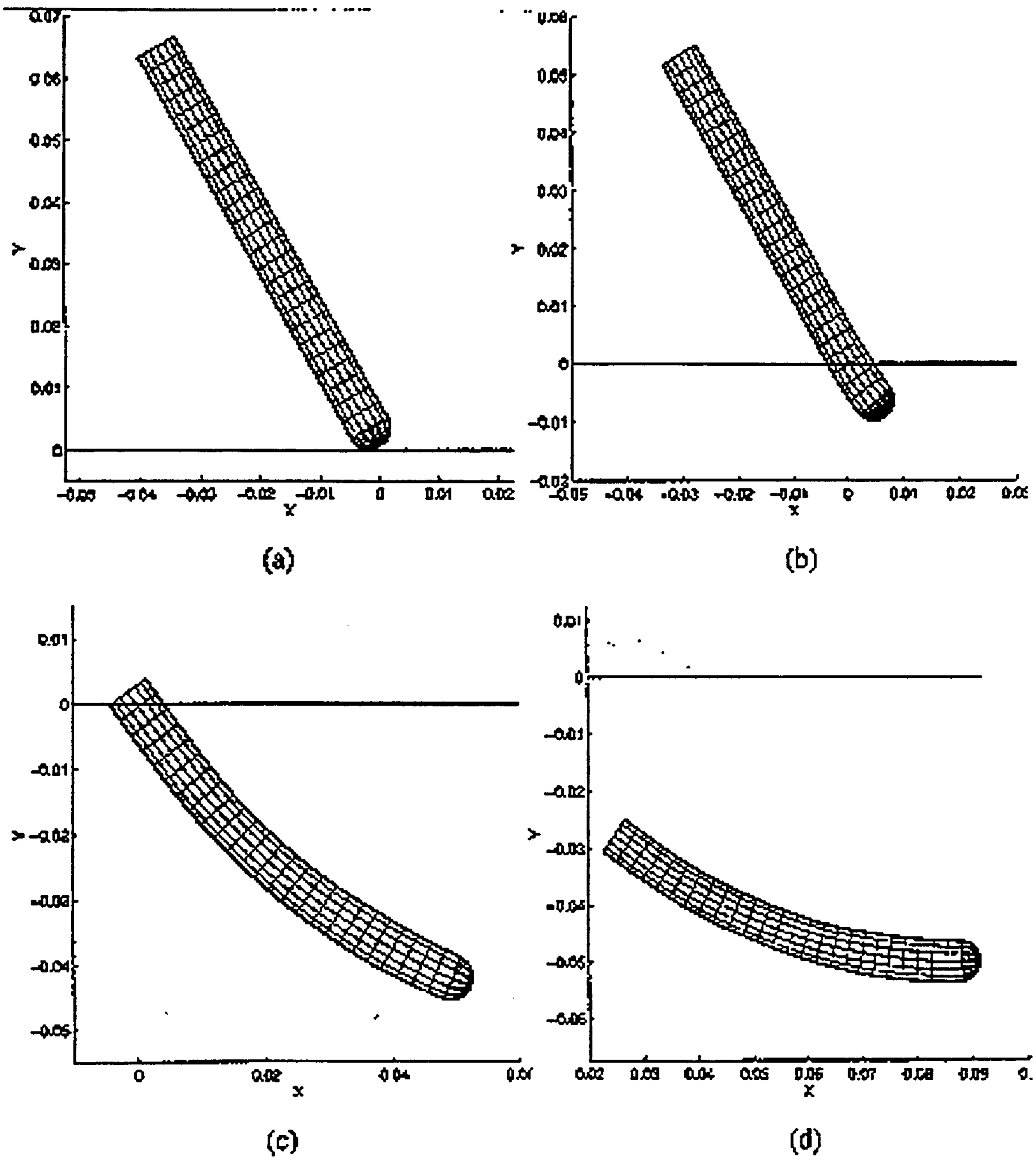


Figure 5 Projectile deformation for oblique impact at a striking velocity V_s = 960 m/s at: (a) 0.0 μ s; (b) 14 μ s; (c) 98 μ s; and (d) 252 μ s.

Finite Element Input Data

<u>ca_al_rigid.i</u>

```
Title
   penetration into aluminum
Material,1,rigid,8000.
   contact modulus = 1
end
Rigid Time Step = 3.5e-7
```

```
Time Step Scale = 1
Bulk Viscosity 0,0
Initial Velocity Material,1 0,-960,0
Cavity Expansion, 100 axis=Y bounds=0,-10
         coef=1.39087e9,8.50144e5,2.54794e3
Termination Time = 0.00035
Plot Time = 1.4e-5
history time = 7.0e-6
Plot Nodal = displ, velocity, mass
Plot Element = eqps, vonmises
Plot History coord=0,0,0 vari=velo comp=y name=a
Plot History coord=0,0,0 vari=disp comp=y name=a
Exit
```

ca al.i

```
Title
   penetration into aluminum
Material, 1, elastic plastic, 8000.
   youngs modulus = 189.0e9
   poissons ratio = 0.3
   yield stress = 2.067e9
   hardening modulus = 1.777e8
   beta = 1.0
end
Time Step Scale = 1
Bulk Viscosity 0,0
Initial Velocity Material,1 0,-960,0
Cavity Expansion, 100 axis=Y bounds=0,-10
         coef=1.39087e9,8.50144e5,2.54794e3
Termination Time = 0.00035
Plot Time = 1.4e-5
history time =7.0e-6
Plot Nodal = displ, velocity, mass
Plot Element = eqps, vonmises
Plot History coord=0,0,0 vari=velo comp=y name=a
Plot History coord=0,0,0 vari=disp comp=y name=a
Exit
```

ca al 30.i

```
Title
   penetration into aluminum
Material, 1, elastic plastic, 8000.
   youngs modulus = 189.0e9
   poissons ratio = 0.3
   yield stress = 2.067e9
   hardening modulus = 1.777e8
   beta = 1.0
end
Time Step Scale = 1
Bulk Viscosity 0,0
Initial Velocity Material,1 480.0,-831.4,0
Cavity Expansion, 100 axis=Y bounds=0,-10
         coef=1.39087e9,8.50144e5,2.54794e3
Termination Time = 0.00035
Plot Time = 1.4e-5
history time = 7.0e-6
Plot Nodal = displ, velocity, mass
Plot Element = eqps, vonmises
Plot History coord=0,0,0 vari=velo comp=y name=a
Plot History coord=0,0,0 vari=disp comp=y name=a
Exit
```

Problem Template

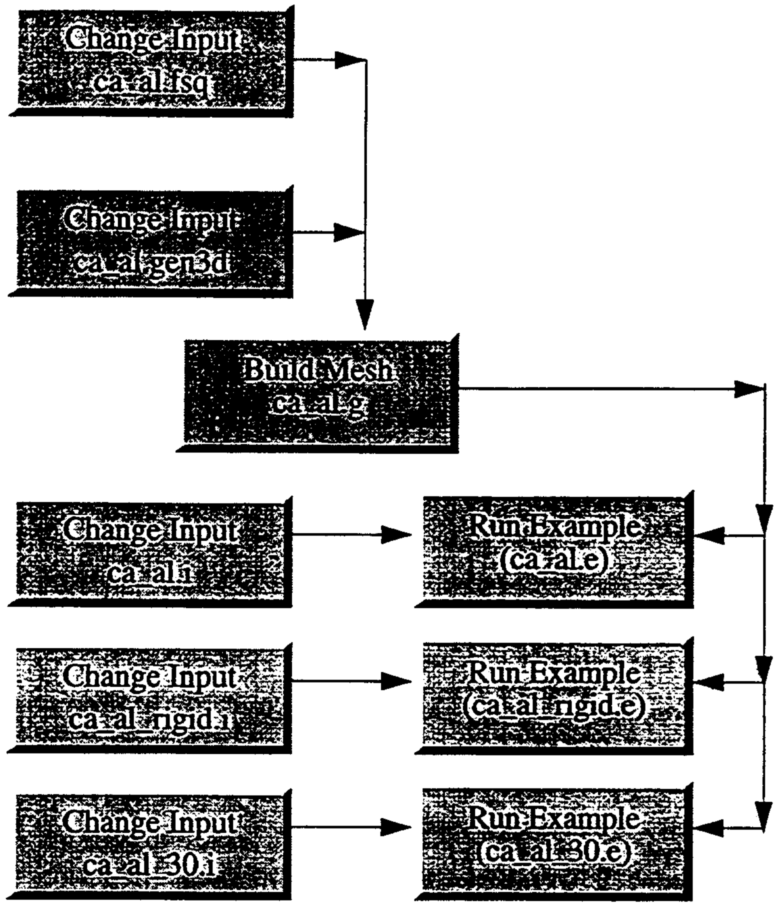


Figure 6 Example template for building the mesh and running the example.

shows an outline of how this problem is constructed using the SEACAS software system. Corresponding text files are included in the Mesh Generation and the Finite Element Input Data sections.

Mesh Generation

The mesh was generated using FASTQ and GEN3D. The following files were used:

```
ca_al.fsq - FASTQ input file
ca_al.gen3d - GEN3D input file
```

The mesh can be made using the Makefile with

```
make ca_al.g
```

The preprocessor APREPRO will evaluate the expressions in the braces '{}'. The impact angle can be changed by resetting the value {ANGLE = 0.0} to {ANGLE=30.0}.

<u>ca_al.fsq</u>

```
{ECHO(OFF)}
{a=7.112e-3/2}
```

```
{CRH=0.5}
{L=71.12e-3}
{s=2*a*CRH}
{l=a*sqrt(4*CRH-1)}
{ECHO(ON)}
 title
  projectile
 point     1     0,0
 point     2     0,{l}
 point     3     {a},{l}
 point     4     0,{L+l}
 point     5     {a},{L+l}
 point     6     {-(s-a)},{l}
 line      1    circ     1 3      6    13   1.
 line      2    str      3 5      0    25   1.
 line      3    str      5 4      0     6   1.
 line      4    str      4 2      0    25   1.
 line      5    str      2 1      0    10   1.
 line      6    str      2 3      0     6   1.
 sidebc 100 1 2
 region     1      1   -6 -2 -3 -4
 region     2      1   -1 -6 -5
 scheme     1 m
 scheme     2 t6s
 body       1 2
 exit
```

cal_al.gen3d

```
{ECHO(OFF)}
{Angle=0.0}
{OffSet=0.0}
{ECHO(ON)}
rotate 16,360
center 1
revolve z,{Angle}
offset 0,{OffSet},0
end
```

Cavity Expansion: Concrete

Keywords cavity expansion, pressure load, penetration

From: [Warren, T.L. and Tabbara, M.R., 1997]

Description

In this example we consider the penetration of 58.4 Mpa (8.5 ksi) concrete targets by solid 3.0 caliber-radius-head (CRH) ogive-nose, 4340 R_c 45 steel rods launched at striking velocities between 400 and 1200 m/s. These rods have density $\rho_p = 7830 \text{ kg/m}^3$, shank length $L = 169.5$ mm, shank diameter $2a = 20.3$ mm, nose length $l = 33.7$ mm, and nominal mass $m = 0.478$ kg. The coefficients for the cavity expansion forcing function are obtained using the semiempirical method developed by [Forrestal, M.J., Altman, B.S., Cargile, J.D., and Hanchak, S.J., 1994] for penetration into concrete. The finite element mesh used in this example is shown in Figure 1 and is comprised of 3,197 nodes and 2,816 elements.

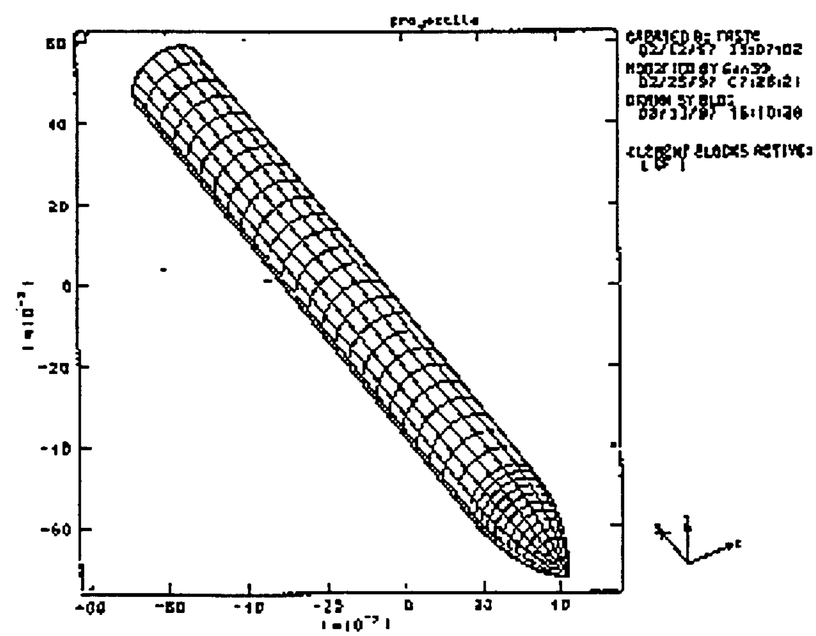


Figure 1 **Finite element mesh of the ogive-nose rod.**

Results and Corroborative Data

As discussed by [Forrestal, M.J., Altman, B.S., Cargile, J.D., and Hanchak, S.J., 1994] there are two regions in the penetration process of concrete. The first region is a conical cratering region with a depth of approximately two projectile diameters. The second region is the tunneling region which starts at the end of the cratering region and proceeds to the final depth of penetration. For the cratering region we average the expression for the stress [Forrestal, M.J., Altman, B.S., Cargile, J.D., and Hanchak, S.J., 1994] acting on the penetrator over the range $0 \le y \le 4a$ which gives:

$$\sigma_r = \frac{1}{4a}\int_0^{4a} \sigma_r dy = \frac{m}{8\pi a^3}\left[V_s^2 - \left(\frac{mV_s^2 - 4\pi a^3 S f'_c}{m + 4\pi a^3 N \rho_0}\right)\right] \tag{1}$$

where m is the mass of the projectile, V_s is the striking velocity, f'_c is the unconfined compressive strength of the target, ρ_0 is the density of the undeformed target material, S is an empirical dimensionless constant, and N is a geometric parameter defined by:

$$N = \frac{8\psi - 1}{24\psi^2} \tag{2}$$

with ψ being the CRH number. Thus, the value of the constant A in the cratering region is directly obtained from the relation (i.e., $B = C = 0$), where we take $Y = f'_c$ and for the given target material $f'_c = 5.84$ MPa, $S = 9.037$, and $\rho_0 = 2320$ kg/m^3. In the tunneling region $A = S$, $B = 0.0$, and $C = 1.0$ [Forrestal, M.J., Altman, B.S., Cargile, J.D., and Hanchak, S.J., 1994].

We assume the 4340 R_c 45 steel projectile to behave as an elastic-plastic power-law hardening material and use the isotropic elastic-plastic power-law hardening material model in PRONTO3D. The density, Young's modulus, yield strength, and Poisson's ratio for 4340 steel projectiles are given by [Luk, V.K. and Piekutowski, A.J., 1991] as $\rho = 7810$ kg/m^3, $E = 206.8$ GPa, $Y = 1.207$ GPa, and $\nu = 0.32$, respectively. Curve fitting the data in [Luk, V.K. and Piekutowski, A.J., 1991] gives a hardening constant of 382 MPa and a hardening exponent of 0.266, which are required values for use in the selected material model. An example input file for a striking velocity of 1162 m/s is shown in ca_concrete.i. Depth of penetration results are compared in Figure 2 with the semiempirical analytical solution of [Forrestal, M.J., Altman, B.S., Cargile, J.D., and Hanchak, S.J., 1994], and with experimental data obtained by [Frew, D.J., Hanchak, S.J., Green, M.L., and Forrestal, M.J., 1997]. Good agreement is observed with both the analytical solution and experimental data. A termination time of 3,000 ms was used for each of the striking velocities, requiring approximately 3,785 cpu seconds on a CRAY J-90.

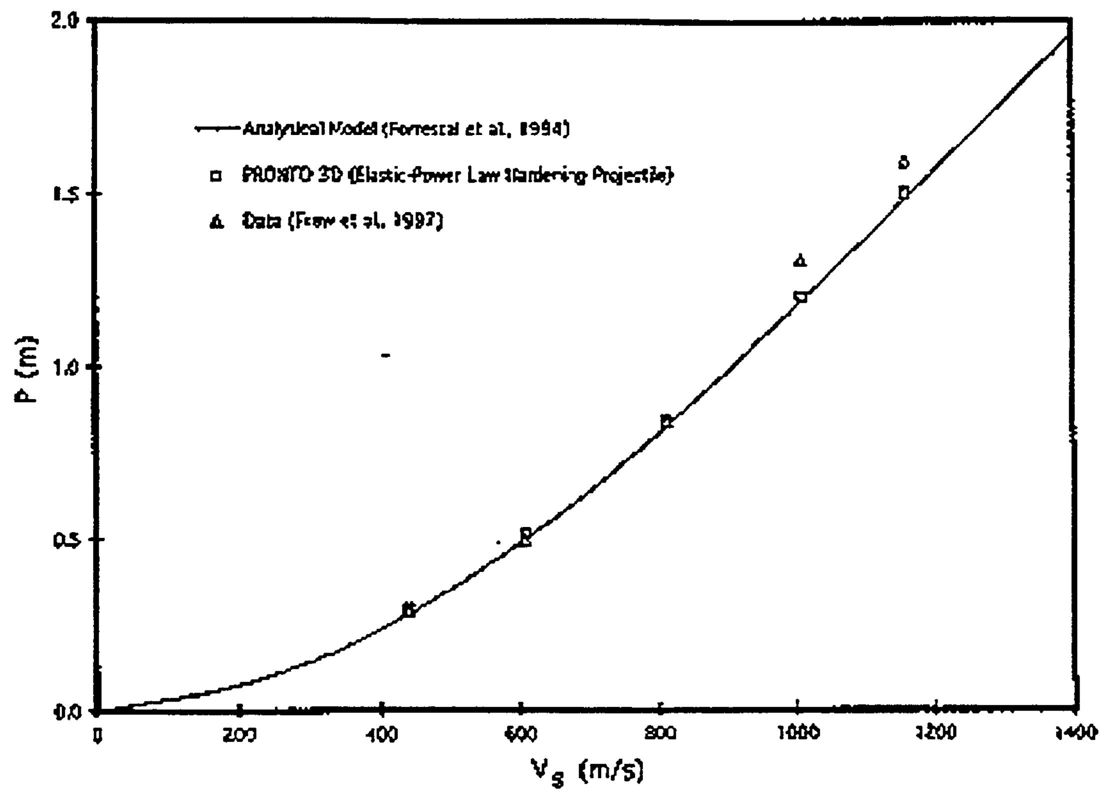


Figure 2 **Depth of penetration versus striking velocity for an elastic plastic projectile.**

Finite Element Input Data

ca concrete.i

```
title
   penetration into concrete
material,1,ep power hard, 7810.
   youngs modulus = 206.8e9
   poissons ratio = 0.32
   yield stress = 1.207e9
   hardening constant = 3.8247155277e8
   hardening exponent = 0.2962824
   luders strain = 0.0
end
bulk viscosity 0,0
initial velocity material,1 0,-1162.0,0
cavity expansion, 100 axis=Y bounds=0,-0.0406 coef=4.27757e8,0,0.0
cavity expansion, 100 axis=Y bounds=-0.0406,-10 coef=5.277e8,0,2.320e3
termination time=3.0e-3
plot time=1.2e-4
plot element = pressure,eqps,vonmises
plot nodal = displacement,velocity,mass
plot history coord=0,0,0 vari=velo comp=y name=a
plot history coord=0,0,0 vari=disp comp=y name=a
history time=6e-5
exit
```

Problem Template

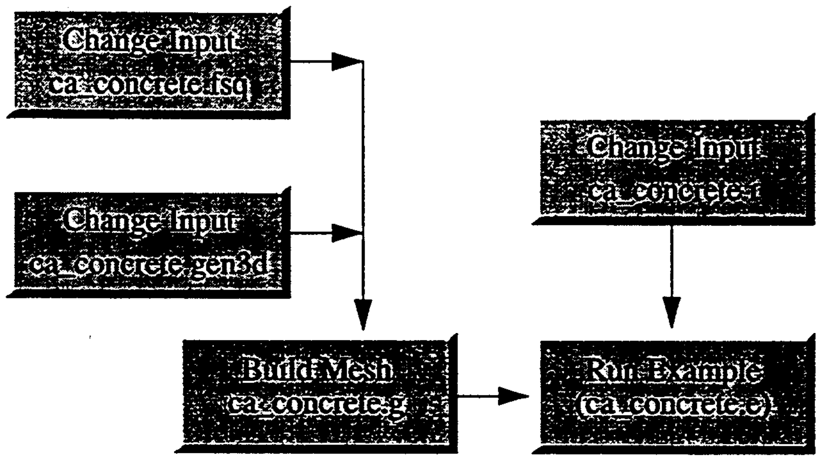


Figure 3 **Example template for building the mesh and running the example.**

Figure 3 shows an outline of how this problem is constructed using the SEACAS software system. Corresponding text files are included in the Mesh Generation and the Finite Element Input Data sections.

Mesh Generation

The mesh was generated using FASTQ and GEN3D. The following files were used:

```
ca_concrete.fsq - FASTQ input file
ca_concrete.gen3d - GEN3D input file
```

The mesh can be made using the Makefile with

```
make ca_concrete.g
```

The pre-processor APREPRO will evaluate the expressions in the braces '{}'. The impact angle can be changed by resetting the value {ANGLE = 0.0} to {ANGLE = 30.0}.

ca_concrete.fsq

```
{ECHO(OFF)}
{a=7.112e-3/2}
{CRH=3.0}
{L=59.33e-3}
{s=2*a*CRH}
{l=a*sqrt(4*CRH-1)}
{ECHO(ON)}
 title
  projectile
 point     1    0,0
 point     2    0,{l}
 point     3    {a},{l}
 point     4    0,{L+l}
 point     5    {a},{L+l}
 point     6    {-(s-a)},{l}
 line      1   circ     1 3     6   13  1.
 line      2   str      3 5     0   25  1.
 line      3   str      5 4     0    6  1.
```

```
line      4   str     4 2     0   25  1.
line      5   str     2 1     0   10  1.
line      6   str     2 3     0    6  1.
sidebc 100 1 2
region      1      1  -6 -2 -3 -4
region      2      1  -1 -6 -5
scheme    1 m
scheme    2 t6s
body      1 2
exit
```

ca_concrete.gen3d

```
{ECHO(OFF)}
{Angle=0.0}
{OffSet=0.0}
{ECHO(ON)}
rotate 16,360
center 1
revolve z,{Angle}
offset 0,{OffSet},0
end
```

Bibliography

Adley, M.D., and Moxley, R.E., 1996 *PENCURV/ABAQUS: A Simply Coupled Penetration Trajectory/Structural Dynamics Model for Deformable Projectiles Impacting Complex Curvilinear Targets*, Technical Report SL-96-6, U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS.

American National Standards Institute, 1978 *American National Standards Programming Language FORTRAN* ANSI X3.9-1978, American National Standards Institute, New York.

American Society for Testing and Materials, 19xx *Annual Book of ASTM Standards: Section 3 - Metals Test Methods and Analytical Procedures*, American Society for Testing and Materials, Philadelphia, Pennsylvania.

Attaway, S.W., 1990 *Update of PRONTO 2D and PRONTO 3D Transient Solid Dynamics Program*, SAND90-0102, Sandia National Laboratories, Albuquerque, NM.

Balmer, H.A. and Witmer, E.A., 1964 *Theoretical-Experimental Correlation of Large Dynamic and Permanent Deformation of Impulsively Loaded Simple Structures*, FDP-TDR-64-108, Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, OH.

Bartels, R. and Daniel, J.W., 1973 "A conjugate gradient approach to nonlinear elliptic boundary value problems in irregular regions," in *Lecture Notes in Mathematics*, (A. Dold and B. Eckmann, eds.), Springer-Verlag, New York, pp. 1-11.

Bathe, K.J. and Wilson, E.L., 1976 *Numerical Methods in Finite Element Analysis*, Prentice-Hall, Inc., New Jersey.

Beisinger, Z.E., 1984 *SEACO: Sandia Engineering Analysis Department Code Output Data Base*, SAND84-2004, Sandia National Laboratories, Albuquerque, NM.

Belytschko, T. and Bindelman, L.P., 1993 "Assumed strain stabilization of the eight node hexahedral element," *Compter Methods in Applied Mechanics and Engineering*, Vol. 105, pp. 225-260.

Belytschko, T. and Lin, J.I., 1985 "Eigenvalues and Stable Time Step for the Bilinear Mindlin Plate Element," *International Journal for Numerical Methods in Engineering*, Vol. 21, pp. 1729-1745.

Belytschko, T. and Lin, J.I., 1987 "A Three-Dimensional Impact-Penetration Algorithm with Erosion," *Computers and Structures*, Vol. 25, No. 1, pp. 95-104.

Belytschko, T., Lin, J.I. and Tsay, C.S., 1984 "Explicit Algorithms for the Nonlinear Dynamics of Shells," *Computer Methods in Applied Mechanics and Engineering*, Vol. 42, pp. 225-251.

Belytschko, T. and Marchertas, A.H., 1974 "Nonlinear Finite Element Method for Plates and its Application to the Dynamic Response of Reactor Fuel Subassemblies," *Journal of Pressure Vessel Technology*, ASME, pp. 251-157.

Belytschko, T. and Neal, M.O., 1991 "Contact-Impact by the Pinball Algorithm with Penalty and Lagrangian Methods," *International Journal Numerical Methods of Engineering*, Vol. 31, pp. 547-572.

Belytschko, T., Ong, J.S.-J., Liu, W.K. and Kennedy, J.M., 1984 "Hourglass Control in Linear and Nonlinear Problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 43, pp. 251-276.

Belytschko, T., Schwer, L. and Klein, M.J., 1977 "Large Displacement Transient Analysis of Space Frames," *International Journal for Numerical Methods in Engineering*, Vol. 11, pp. 65-84.

Benson, D.J. and Hallquist, J.O., 1990 "A Single Surface Contact Algorithm for the Post-Buckling Analysis of Structures," *Computer Methods in Applied Mechanics and Engineering*, Vol. 78, pp. 141-163.

Benz, W., 1988 "Applications of Smooth Particle Hydrodynamics (SPH) to Astrophysical Problems," *Computational Physics Commuinications*, Vol. 48, pp. 97-105.

Benz, W., 1990 "Smooth Particle Hydrodynamics: A Review," in *The Numerical Modeling of Stellar Pulsation*, (J.R. Buchler, ed.), Kluwer, Dordrecht, Netherlands, p. 269.

Bergmann, V.L., 1991 *Transient Dynamic Analysis of Plates and Shells with PRONTO 3D*, SAND91-1182, Sandia National Laboratories, Albuquerque, NM.

Bertsekas, D.P., 1982 *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York.

Biffle, J.H., 1981 *JAC--A Two-Dimensional Finite Element Computer Program for the Nonlinear Response of Solids with the Conjugate Gradient Method*, SAND81-0998, Sandia National Laboratories, Albuquerque, NM.

Biffle, J.H., 1984 *JAC - A Two-Dimensional Finite Element Computer Program for the Nonlinear Quasi-Static Response of Solids with the Conjugate Gradient Method*, SAND81-0998, Sandia National Laboratories, Albuquerque, NM.

Biffle, J.H., 1993 *JAC3D - A Three-Dimensional Finite Element Computer Program for the Nonlinear Quasi-Static Response of Solids with the Conjugate Gradient Method*, SAND87-1305, Sandia National Laboratories, Albuquerque, NM.

Biffle, J.H. and Gubbels, M.H., 1976 *WULFF - A Set of Computer Programs for the Large Displacement Dynamic Response of Three Dimensional Solids*, SAND76-0096, Sandia National Laboratories, Albuquerque, NM.

Biffle, J.H. and Stone, C.M., 1989 "Personal communications," Applied Mechanics Division, Sandia National Laboratories, Albuquerque, NM.

Bishop, R.F., Hill, R., and Mott, N.F., 1945 "The theory of indentation and hardness", *Proceedings of the Royal Society*, Vol. 57 (3), pp. 147-159.

Blacker, T.D., 1988 *FASTQ Users Manual Version 1.2*, SAND88-1326, Sandia National Laboratories, Albuquerque, NM.

Boaz, M.L., 1966 *Mathematical Methods in the Physical Sciences*, John Wiley and Sons, New York, NY.

Budiansky, B. and O'Connell, R.J., 1976 "Elastic moduli of a cracked solid," *Computer Methods in Applied Mechanics and Engineering*, Vol. 12, pp. 81-97.

CRAY Research, Inc., 1989 *Volume 3: UNICOS Math and Scientific Library Reference Manual*, SR-2081 5.0.

Campbell, P.M., 1988 *Some New Algorithms for Boundary Values Problems in Smooth Particle Hydrodynamics*, DNA-TR-88-286.

Carpenter, N.J., Taylor, R.L. and Katona, M.G., 1991 "Lagrange Constraints for Transient Finite Element Surface Contact," *International Journal for Numerical Methods in Engineering*, Vol. 32, 103-128.

Chait, R., 1972 "Factors influencing the strength differential of high strength steels", *Metalurgical Transactions A*, Vol. 3, pp. 365-371.

Chakarabarty, J., 1987 *Theory of Plasticity*, McGraw-Hill Book Company, New York, pp. 306-315, 342-350.

Chaudhary, A.B. and Bathe, K.J., 1986 "A Solution Method for Static and Dynamic Analysis of Three-Dimensional Contact Problems with Friction," *Computers and Structures*, Vol. 24, No. 6, pp. 855-873.

Cheung, C.Y. and Cebon, D., 1997 "Experimental Study of Pure Bitumens in Tension, Compression, and Shear," *Journal of Rheology*, Vol. 41, No. 1, pp. 45-73.

Chou, T.S., 1989 *The Dynamic Response at High-Explosive Inert-Solid Interface as Predicted by Finite Difference/Element Computer Programs*, Tech. Rep. MLM-3562, EG&G Mound Applied Technologies, Miamisburg, OH.

Cloutman, L.D., 1990(a) *Basics of Smoothed Particle Hydrodynamics*, Lawrence Livermore National Laboratory, Livermore, CA, report UCRL-ID-103698.

Cloutman, L.D., 1990(b) "An Evaluation of Smoothed Particle Hydrodynamics," *Proceedings of The NEXT Free-Lagrange Conference*, Jackson Lake Lodge, Moran, WY, June 3-7.

Cohen, M. and Jennings, P.C., 1983 "Silent Boundary Methods," in *Computational Methods for Transient Analysis*, (Belytschko, T. and Hughes, T.J.R., eds.), North-Holland.

Cole, R.H., 1965 *UnderWater Explosions*, Dover Publications.

Concus, P., Golub, G.H. and O'Leary, D.P., 1976 "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations," in *Sparse Matrix Computations*, (J.R. Bunch and D.J. Rose, eds.), Academic Press, New York, pp. 309-332.

Courant, R., Friedrichs, K.O. and Lewy, H., 1928 *Mathematical Annotations*, Vol. 100, p. 32.

Crismann, J.M. and Zapas, L.J., 1979 "Creep Failure and Fracture of Polyethylene in Uniaxial Extension," *Polymer Engineering Science*, Vol. 19, pp. 99-103.

Curnier, A. and Alart, P., 1988 "A Generalized Newton Method for Contact Problems with Friction," *Journal de Mecanique Theorique et Appliquee*, Vol. 7, 67-82.

Daniel, J.W., 1967 "The conjugate gradient method for linear and nonlinear operator equations," *SIAM Journal of Numerical Analysis*, Vol. 4, no. 1, pp. 10-26.

Dennis, J.E. and Schnabel, R.B., 1996 *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Society for Industrial and Applied Mathematics, Philadelphia.

Dienes, J.K., 1979 "On the analysis of rotation and stress rate in deforming bodies," *Acta Mechanica*, Vol. 32, pp. 217-232.

Dobratz, B.M., 1981 *LLNL Explosives Handbook, Properties of Chemical Explosives and Explosive Simulants*, UCRL-52997, Lawrence Livermore National Laboratory, Livermore, CA.

Duffey, T.A., and Macek, R.W., 1997 "Non-normal impact of earth penetrators", *Proceedings of the International Symposium on Penetration and Impact Problems* (ICES'97), San Jose, Costa Rica.

Engleman, R. and Jaeger, Z., 1987 *Theoretical Aids for Improvement of Blasting Efficiencies in Oil Shale and Rocks*, Tech. Rep AP-TR-12/87, Soreq Nuclear Research Center, Yavne, Israel.

Ferry, 1950 ???

Flanagan, D.P. and Belytschko, T., 1981 "A Uniform Strain Hexahedron and Quadrilateral with Orthogonal Hourglass Control," *International Journal for Numerical Methods in Engineering*, Vol. 17, pp. 679-607.

Flanagan, D.P. and Belytschko, T., 1984 "Eigenvalues and Stable Time Steps for the Uniform Hexahedron and Quadrilateral," *Journal of Applied Mechanics*, Vol. 51, pp. 35-40.

Flanagan, D.P., Mills-Curran, W.C., and Taylor, L.M., 1986 *SUPES - A Software Utilities Package for the Engineering Sciences*, SAND86-0911, Sandia National Laboratories, Albuquerque, NM.

Flanagan, D.P. and Taylor, L.M., 1987 "An accurate numerical algorithm for stress integration with finite rotations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 62, pp. 305-320.

Flanagan, D.P. and Taylor, L.M., 1987 "On the Analysis of Rotation and Stress Rate in Deforming Bodies," *Computer Methods in Applied Mechanics and Engineering*, Vol. 62, pp. 305-320.

Fletcher, R. and Reeves, C.M., 1964 "Function minimization by conjugate gradients," *The Computer Journal*, Vol. 7, pp. 149-154.

Forrestal, M.J., Altman, B.S., Cargile, J.D., and Hanchak, S.J., 1994 "An empirical equation for penetration depth of ogive-nose projectiles into concrete targets", *International Journal of Impact Engineering*, Vol. 15, pp. 395-405.

Forrestal, M.J., Brar, N.S., and Luk, V.K., 1991 "Penetration of strain-hardening targets with rigid spherical-nose rods", *ASME Journal of Applied Mechanics*, Vol. 58, pp. 7-10.

Forrestal, M.J. and Luk, V.K., 1991 "Penetration into soil targets", *International Journal of Impact Engineering*, Vol. 12, pp. 427-444.

Forrestal, M.J., Okajima, K., and Luk, V.K., 1988 "Penetration of 6061-T651 aluminum targets with rigid long rods", *ASME Journal of Applied Mechanics*, Vol. 55, pp. 755-760.

Forrestal, M.J., and Tzou, D.Y., 1996 "A spherical cavity-expansion penetration model for concrete targets", *International Journal of Solids Structures* (accepted).

Forrestal, M.J., Tzou, D.Y., Askari, E., and Longcope, D.B., 1995 "Penetration into ductile metal targets with rigid spherical-nose rods", *International Journal of Impact Engineering*, Vol. 16, pp. 699-710.

Freudenthal, 1963 ???

Frew, D.J., Hanchak, S.J., Green, M.L., and Forrestal, M.J., 1997 "Penetration of concrete targets with ogive-nose steel rods", *International Journal of Impact Engineering*, (submitted).

Frost and Ashby, 19?? *Deformation Mechanisms Maps*, ???.

Fung, Y.C., 1965 *Foundations of Solid Mechanics*, Prentice-Hall, Englewood Cliffs, New Jersey.

Fung, Y.C., 1977 *A First Course in Continuum Mechanics*, 2nd edition, Prentice-Hall, Englewood Cliffs, New Jersey.

Gallego, F.J. and Anza, J.J., 1989 "A Mixed Finite Element Model for the Elastic Contact Problem," *International Journal for Numerical Methods in Engineering*, Vol. 28, 1249-1264.

Giannakopoulos, A.E., 1989 "The Return Mapping Method for the Integration of Friction Constitutive Relations," *Computers and Structures*, Vol. 32, 157-167.

Gilkey, A.P., 1988 *ALGEBRA - A Program that Algebraically Manipulates the Output of a Finite Element Analysis (EXODUS Version)*, SAND88-1431, Sandia National Laboratories, Albuquerque, NM.

Gilkey, A.P. and Glick, J.H., 1989 *BLOT - A Mesh and Curve Plot Program for the Output of a Finite Element Analysis*, SAND88-1432, Sandia National Laboratories, Albuquerque, NM.

Gilkey, A.P. and Sjaardema, G.D., 1989 *GEN3D: A GENESIS Database 2D to 3D Transformation Program*, SAND89-0485, Sandia National Laboratories, Albuquerque, NM.

Gingold, R.A. and Monaghan, J.J., 1982 "Kernel Estimates as a Basis for General Particle Methods in Hydrodynamics," *Journal Computational Physics*, Vol. 46, pp. 429-453.

Glowinski, R. and LeTallec, P., 1989 *Augmented Lagrangian and Operator Splitting Methods in Nonlinear Mechanics*, SIAM, Philadelphia.

Goodier, J.N. 1965 "On the mechanics of indentation and cratering in the solid targets of strain-hardening metal by impact of hard and soft spheres", *Proceedings of the 7th Symposium on Hypervelocity Impact III*, pp. 215-259.

Grady, D., 1983 "The Mechanics of Fracture Under High-rate Stress Loading," in *William Prager Syposium on Mechanics of Geomaterials: Rocks, Concretes and Soils*, (Bazant, Z.P., ed.).

Guenther, C., Hicks, D.L. and Swegle, J.W., 1994 *Conservative Smoothing versus Artificial Viscosity*, SAND94-1853, Sandia National Laboratories, Albuquerque, NM.

Gurtin, M.E., 1981 *An Introduction to Continuum Mechanics*, Academic Press, Inc.

Hallquist, J.O., 1981 *User's Manual for DYNA3D and DYNAP*, Lawrence Livermore National Laboratory, Livermore, CA.

Hallquist, J.O., 1982 *User's Manual for DYNA2D - An Explicit Two-Dimensional Hydrodynamic Finite Element Code with Interactive Rezoning*, Lawrence Livermore National Laboratory, Livermore, CA.

Hallquist, J.O., 1984 *NIKE3D: An Implicit, Finite Deformation, Finite Element Code for Analyzing the Static and Dynamic Response of Three Dimensional Solids*, UCID-18822, Lawrence Livermore National Laboratory, Livermore, CA.

Hallquist, J.O., 1984 *User's Manual for DYNA2D: An Explicit Two-Dimensional Hydrodynamic Finite Element Code with Interactive Rezoning*, Rev. 2, UCID-18756, Lawrence Livermore National Laboratory, Livermore, CA.

Hallquist, J.O., 1986 *NIKE2D: A Vectorized Implicit, Finite Deformation Finite Element Code for Analyzing the Static and Dynamic Response of 2-D Solids with Interactive Rezoning and Graphics*, UCID-19677, Lawrence Livermore National Laboratory, Livermore, CA.

Hallquist, J.O., Goudreau, G.L. and Benson, D.J., 1985 "Sliding Interfaces with Contact-Impact in Large-Scale Lagrangian Computations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 51, 107-137.

Hallquist, J.O. and Benson, D.J., 1986 "A comparison of an Implicit and Explicit Implementation of the Hughes-Liu Shell," in *Finite Element Methods for Plate and Shell Structures, Volume 1: Element Technology*, (Hughes, T.J.R. and Hinton, E., eds.), Pineridge Press International, Swansea, U.K., pp. 394-430.

Hallquist, J.O. and Benson, D.J., 1987 *User's Manual for DYNA3D: Nonlinear Dynamic Analysis of Structures*, Rev. 3, UCID-19592, Lawrence Livermore National Laboratory, Livermore, CA.

Harding, D.C., Attaway, S.W., Neilsen, J., Blacker, T.D. and Pierce, J., 1992 *Evaluation of Four Multiple Package Crush Environment to the Common Package, Model 1, Plutonium Air Transport Container*, SAND92-0278, Sandia National Laboratories, Albuquerque, NM.

Harlow, F.H., 1988 "PIC and its Progeny," *Computational Physics Communications*, Vol. 48, pp. 1-10.

Hearmon, R. F. S., 1961 *An Introduction to Applied Anisotropic Elasticity*, Oxford University Press, Oxford, England.

Herrman and Peterson, 1968 ???

Heinstein, M.W., Attaway, S.W., Mellow, F.J. and Swegle, J.W., 1993 *A General-Purpose Contact Detection Algorithm for Nonlinear Structural Analysis Codes*, SAND92-2141, Sandia National Laboratories, Albuquerque, NM.

Hestenes, M.R. and Stiefel, E., 1952 "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, Vol. 49, pp. 409-436.

Hibbitt, Karlsson and Sorensen, Inc., 1989 *ABAQUS Users Manual, Version 4-8*, Hibbitt, Karlsson and Sorensen, Inc., Providence, Rhode Island.

Hibbitt, Karlsson and Sorensen, Inc., 1992 *Contact Calculations with ABAQUS - ABAQUS Explicit Users Manual*, Hibbitt, Karlsson and Sorensen, Inc., Providence, Rhode Island.

Hicks, D.L., 1978 "Stability Analysis of WONDY (A Hydrocode Based on the Artificial Viscosity Method of von Neumann and Richtmyer) for a Special Case of Maxwell's Law," *Mathematics of Computation*, Vol. 32, pp. 1123-1130.

Hilber, Hughes, and Taylor, 1977 "Improved Numerical Dissipation for Time Integration Algorithms in Structural Dynamics," *Earthquake Engineering and Structural Dynamics*, Vol. 5, pp.283-292.

Hill, R., 1948 *A Theory of Earth Movement Near a Deep Underground Explosion*, Memo No. 21-48, Armament Research Establishment, Fort Halstead, Kent, UK.

Hill, R., 1950 *The Mathematical Theory of Plasticity*, Oxford University Press, London.

Holcomb, D.J., 1985 "Results of the 55 Day Consolidation Test," Internal Memorandum to J. Stormont, 6332, Sandia National Labs, Albuquerque, NM.

Holcomb, D.L and Hannum, D.W., 1982 *Consolidation of Crushed Salt Backfill Under Conditions Applicable to the WIPP Facility*, SAND82-0630, Sandia National Labs, Albuquerque, NM.

Holcomb, D.J. and Shields, M.F., 1987 *Hydrostatic Consolidation of Crushed Salt with Added Water*, SAND87-1990, Sandia National Labs, Albuquerque, NM.

Holden, J.T., 1972 "On the finite deflections of thin beams," *International Journal of Solids and Structures*, Vol. 8, pp. 1051-1055.

Hopkins, H.G., 1960 "Dynamic expansion of spherical cavities in metals", *Progress in Solid Mechanics*, Vol. 1, (Editors I. Sneddon and R. Hill), North Holland, New York, pp. 85-164.

Huang, J., 1969 "Transient Interaction of Plane Acoustic Waves with a Spherical Elastic Shell," *Journal Acoustical Society of America*, Vol. 45, pp. 661-670.

Hughes, T.J.R., 1987 *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Hughes, T.J.R. and Winget, J., 1980 "Finite rotation effects in numerical integration of rate constitutive equations arising in large-deformation analysis," *International Journal for Numerical Methods in Engineering,* Vol. 15, No. 12, pp. 1862-1867.

Ide,Y. and White, J.L., 1977 "Investigation of Failure During Elongational Flow of Polymer Melts," *Journal of Non-Newtion Fluid Mechanics*, Vol. 2, pp. 281-298.

Ide,Y. and White, J.L., 1979 "Experimental Study of Elongational Flow and Failure of Polymer Melts," *Journal of Applied Polymer Science*,Vol. 22, pp. 1061-1079.

International Nickel Company, Inc., The, 1964 *18% Nickel Maraging Steels*, New York, NY.

Irons, B. and Elsawaf, A., 1977 "The conjugate Newton algorithm for solving finite element equations," *Formulations and Computation Algorithms in Finite Element Analysis*, (K.J. Bathe and W. Wunderlich, eds.), MIT Press, pp. 655-672.

Johnson, G.C. and Bammann, D.J., 1984 "A discussion of stress rates in finite deformation bodies," *International Journal of Solids and Structures*, Vol. 20, no. 8, pp. 725-737.

Johnson, G.C. and Bammann, D.J., 1984 "On the Analysis of Rotation and Stress Rate in Deforming Bodies," *International Journal of Solids and Structures*, Vol. 20, No. 8, pp. 725-737.

Johnson, G.R., 1983 *Development of Strength and Fracture Models for Computations Involving Severe Dynamic Loading, Vol. I: Strength and Fracture Models*, Tech. Rep. AFATL-TR-83-05, Air Force Armament Laboratory.

Johnson, G.R., 1988 "Implementation of simplified constitutive models in large computer codes," in *Dynamic Constitutive/Failure Models*, (A. M. Rajendran and T. Nichols, eds.), pp. 409-426, Dayton, OH.

Johnson, G.R. and Holmquist, T.J., 1989 *Test Data and Computational Strength and Fracture Model Constants for 23 Materials Subjected to Large Strains, High Strain Rates, and High Temperatures*, Tech. Rep. LA-11463-MS, Los Alamos National Laboratory, Los Alamos, NM.

Johnson, G.R., Stryk, R.A., Holmquist, T.J., and Beissel, S.R., 1996 *User Instructions for the 1996 version of the EPIC Code*, Alliant Techsystems Inc., Hopkins, MN.

Johnson, G.R., Stryk, R.A., Holmquist, T.J., and Beissel, S.R., 1997 *EPIC 97* (software to be released), Alliant Techsystems Inc., Hopkins, MN.

Johnson, J.B., 1989 "Personal communication," USACRREL, Building 4070, Ft. Wainwrite, Alaska.

Ju, J.-W. and Taylor, R.L., 1988 "A Perturbed Lagrangian Formulation for the Finite Element Solution of Nonlinear Frictional Contact Problems," *Journal of Theoretical and Applied Mechanics*, Vol. 7, 1-14.

Kamei, E. and Onogi, 1975 "Extensional and Fractural Properties of Monodisperse Polystyrenes at Elevated Temperatures," *Applied Polymer Symposium*, Vol 27, pp. 19-46.

Kaplan, W., 1959 *Advanced Calculus*, Addison-Wesley Publishing, Reading MA.

Kelley, C.T., 1995 *Iterative Methods for Linear and Nonlinear Equations*, Society for Industrial and Applied Mathematics, Philadelphia.

Kerighan, B.W., and Ritchie, D.M., 1978 *The C Programming Language*, Prentice-Hall, Inc., New Jersey.

Key, S.W., Beisinger, Z.E. and Kreig, R.D., 1978 *HONDO II -A Finite Element Computer Program for the Large Deformation Dynamics Response of Axisymmetric Solids*, SAND78-0422, Sandia National Laboratories, Albuquerque, NM.

Kikuchi, N. and Song, Y.J., 1981 "Penalty/Finite Element Approximations of a Class of Unilateral Problems in Linear Elasticity," *Quarterly of Applied Mathematics*, Vol. 39, 1-22.

Kikuchi, N. and Oden, J.T., 1988 *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*, SIAM, Philadelphia.

Kipp, M.E. and Grady, D.E., 1978 *Numerical Studies of Rock Fragmentations*, SAND79-1582, Sandia National Laboratories, Albuquerque, NM.

Kipp, M.E., Grady, D.E. and Chen, E.P., 1980 "Strain-rate Dependent Fracture Initiation," *International Journal of Fracture*, Vol. 16, pp. 471-478.

Kipp, M.E. and Lawrence, R.J., 1982 *WONDY V - A One-Dimensional Finite-Difference Wave Propagation Code*, SAND81-0930, Sandia National Laboratories, Albuquerque, NM.

Krieg, R.D., 1978 *A Simple Constitutive Description for Soils and Crushable Foams*, SC-DR-72-0883, Sandia National Laboratories, Albuquerque, NM.

Krieg, R.D. and Key, S.W., 1976 "Implementation of a Time Dependent Plasticity Theory into Structural Computer Programs," *Constitutive Equations in Viscoplasticity: Computational and Engineering Aspects*, Vol. 20, ASME, New York, p. 125.

Kreig, R.D. and Krieg, D.B., 1977 "Accuracies of numerical solution methods for the elastic-perfectly plastic model," *ASME Journal of Pressure Vessel Technology*, Vol. 99, pp. 510-515.

Kuszmaul, J.S., 1987 "A New Constitutive Model for Fragmentation of Rock Under Dynamic Loading," in *Proceedings of the Second International Symposium on Fragmentation by Blasting*, pp. 412-423, Keystone, CO.

Kuszmaul, J.S., 1987 "A Technique for Predicting Fragmentation and Fragment Sizes Resulting from Rock Blasting," in *Proceedings of the 28th U.S., Symposium on Rock Mechanics*, Tucson, AZ.

Lai, W.M, Rubin, D. and Krempl, E., 1993 *Introduction to Continuum Mechanics*, 3rd edition, Pergamon Press.

Laursen, T.A., 1994 "The Convected Description in Large Deformation Frictional Contact Problems," *International Journal of Solids and Structures*, Vol. 31, pp. 669-681.

Laursen, T.A. and Simo, J.C., 1993 "A Continuum-Based Finite Element Formulation for the Implicit Solution of Multibody, Large Deformation Frictional Contact Problems," *International Journal for Numerical Methods in Engineering*, Vol. 36, 3451-3485.

Laursen, T.A. and Simo, J.C., 1993 "Algorithmic Symmetrization of Coulomb Frictional Problems Using Augmented Lagrangians," *Computer Methods in Applied Mechanics and Engineering*, Vol. 108, 133-146.

Leaderman, 1943 ???

Lee, E., Finger, M. and Collins, W., 1973 *JWL Equation of State Coefficients for High Explosives*, UCID-16189, Lawrence Livermore National Laboratory, Livermore, CA.

Lee, Radok, and Woodward, 1959 ???

Lenard, M.L., 1976 "Convergence conditions for restarted conjugate gradient methods with inaccurate line searches," *Mathematical Programming*, Vol. 10, pp. 32-51.

Libersky, L. and Petschek, A.G., 1990 "Smooth Particle Hydrodynamics with Strength of Materials," *Proceedings of The NEXT Free-Lagrange Conference*, Jackson Lake Lodge, Moran, WY, June 3-7.

Longcope, D.B., 1991 *Coupled Bending/Lateral Load Modeling of Earth Penetrators*, SAND90-0789, Sandia National Laboratories, Albuquerque, NM.

Longcope, D.B., 1996 *Oblique Penetration Modeling and Correlation with Field Tests into a Soil Target*, SAND96-2239, Sandia National Laboratories, Albuquerque, NM.

Lovejoy, S.C. and Whirley, R.G., 1990 *DYNA3D Example Problem Manual*, UCRL-MA-105259, Lawrence Livermore National Laboratory, Livermore, CA.

Luenberger, D.G., 1984 *Linear and Nonlinear Programming*, 2nd ed., Addison-Wesley, Reading, MA.

Lucy, L.B., 1977 "A Numerical Approach to the Testing of the Fission Hypothesis," *Astrophysics Journal*, Vol. 82, pp. 1013-1024.

Luk, V.K. and Piekutowski, A.J., 1991 "An analytical model on penetration of eroding long rods into metallic targets," *International Journal of Impact Engineering*, Vol. 11, pp. 323-340.

Lysmer, J. and Kuhlemeyer, R.L., 1979 "Finite Dynamic Model for Infinite Media," *Journal of the Engineering Mechanics Division of ASCE*, pp. 859-877.

Malkin, A. Y. and Petrie, C.J.S., 1997 "Some Conditions for Rupture of Polymer Liquids in Extension," *Journal of Rheology*, Vol. 41 (1), pp. 1-25.

Malvern, L.E., 1969 *Introduction to the Mechanics of a Continuous Medium*, Prentice-Hall, Inc., New Jersey, pp. 226-228.

Marsden, J.E. and Hughes, T.J.R., 1983 *Mathematical Foundations of Elasticity*, Prentice-Hall, Englewood Cliffs, New Jersey.

Matthies, H. and Strang, G., 1979 "The Solution of Nonlinear Finite Element Equations," *International Journal for Numerical Methods in Engineering*, Vol. 14, 1613-1626.

McClelland, 19?? ???

McClure, C., 1993 *Preliminary Report on Explosive Field Tests in Support of the Hull Deformation/Rupture Study*, NSWC Report NSWCDD/TN-93/94.

Mendelson, A., 1968 *Plasticity: Theory and Application*, The Macmillan Company, New York, pp. 138-156.

Metzinger, K., Attaway, S. and Mello, F., 1991 "Bobbin Stresses Generated by Wire Winding," *First International Conference of Web Handling*, Oklahoma State University, Stillwater, OK.

Michalowski, R. and Mroz, Z., 1978 "Associated and Non-Associated Sliding Rules in Contact Friction Problems," *Archives of Mechanics*, Vol. 30, 259-276.

Mills-Curran, W.C., 1988 *EXODUS: A Finite Element File Format for Pre- and Post-Processing*, SAND87-2997, Sandia National Laboratories, Albuquerque, NM.

Mindlin, R.D., 1951 "Influence of Rotatory Inertia and Shear on Flexural Motions of Isotropic, Elastic Plates," *Journal of Applied Mechanics*, Vol. 18, pp. 31-38.

Monaghan, J.J., 1982 "Why Particle Methods Work," *SIAM Journal Scientific Statistical Computing*, Vol. 3, pp. 422-433.

Monaghan, J.J., 1985 "Particle Methods for Hydrodynamics," *Computational Physics Reports*, Vol. 3, pp. 71-124.

Monaghan, J.J., 1988 "An Introduction to SPH," *Computational Physics Communications*, Vol. 48, pp. 89-96.

Monaghan, J.J. and Gingold, R.A., 1983 "Shock Simulation by the Particle Method SPH," *Journal of Computational Physics*, Vol. 52, pp. 374-389.

Morland and Lee, 1960 ???

Mungiza, A., Owen, D.R.J. and Bicanic, N., 1995 "A Combined Finite-Discrete Element Method in Transient Dynamics of Fracturing Solids," *Engineering Computations*, Vol. 12, pp. 145-174.

Muki and Sternberg, 1961 ???

Neilsen, M.K., Morgan, H.S. and Krieg, R.D., 1986 *A Phenomenological Constitutive Model for Low Density Polyurethane Foams*, SAND86-2927, Sandia National Laboratories, Albuquerque, NM.

Nemat-Nasser, S., Chung, D. and Taylor, L.M., 1989 "Phenomenological modeling of rate-dependent plasticity for high strain rate problems," *Mechanics of Materials*, Vol. 7, No. 4, pp. 319-344.

Newmark, N.M., 1959 "A Method of Computation for Structural Dynamics," *Journal of the Engineering Mechanics Division*, ASCE, 67-94.

Ogden, R.W., 1987 "Recent advances in the phenomenological theory of rubber elasticity," *Rubber Chemistry and Technology*, Vol. 59, pp. 361-383.

Parisch, H., 1989 "A Consistent Tangent Stiffness Matrix for Three Dimensional Non-Linear Contact Analysis," *International Journal for Numerical Methods in Engineering*, Vol. 28, 1803-1812.

Patel, N.R. and Finnie, I., 1969 *???*, Report UCRL-13420, Lawrence Livermore Laboratory, Livermore, CA.

Pearson, G.H. and Connelly, R.W., 1982 "The Use of Extensional Rheometry to Establish Operating Parameters for Stretching Processes," *Journal of Applied Polymer Science*, Vol. 27, pp. 969-981.

Perzyna, P., 1966 "Fundamental Problems in Viscoplasticity," in *Recent Advances in Applied Mechanics*, Academic Press, New York, pp. 243-377.

Pfeiffle, T.W. and Senseny, P.E., 1985 *Permeability and Consolidation of Crushed Salt from the WIPP Site*, Topical Report RSI-0278, RE/SPEC Inc., Rapid City, SD.

Pilkey, W.D. and Wunderlich, W., 1994 *Mechanics of Structures: Variational and Computational Methods*, CRC Press.

Plimpton, S.J., 1990 "Molecular Dynamics Simulations of Short-Range Force Systems on 1024-Node Hypercubes," *Proceedings of the Fifth Distributed Memory Computing Conference*, Charleston, SC.

Powell, M.J.D., 1977 "Restart procedures for the conjugate gradient method," *Mathematical Programming*, Vol. 12, pp. 242-254.

Ralston, A. and Wilf, H.S., 1960 *Mathematical Methods of Digital Computers*, John Wiley and Sons Inc., New York, pp. 62-72.

Ratigan and Wagner, 19?? ???

Rebelo, N., Nagtegaal, J.C. and Hibbitt, H.D., 1990 "Finite Element Analysis of Sheet Forming Processes," *International Journal for Numerical Methods in Engineering*, Vol. 30, 1739-1758.

Reedy, J.E.D., 1990 "Memo: Code corrections to pronto's soils and crushable foams model," Applied Mechanics Division, Sandia National Laboratories, Albuquerque, NM.

Richtmyer, R.D. and Morton, K.W., 1967 *Difference Methods for Initial Value Problems*, Interscience, New York.

Rivlin, R.S., 1948 "???", *Philosophical Transactions of the Royal Society of London*, A, pp. 459-490.

Schoof, L.A. and Yarberry, V.R., 1994 *EXODUS II: A Finite Element Data Model*, SAND92-2137, Sandia National Laboratories, Albuquerque, NM.

Schreyer, H.L., Kulak, R.F. and Kramer, J.M., 1979 "Accurate numerical solutions for elastic-plastic models," *ASME Journal of Pressure Vessel Technology*, Vol. 101, pp. 226-234.

Schwarzl and Staverman, 1952 ???

Silling, S. A., 1991 *CTH Reference Manual: Viscoplastic Models*, SAND91-0292, Sandia National Laboratories, Albuquerque, NM.

Simo, J.C., 1992 "Algorithms for Static and Dynamic Multiplicative Plasticity that Preserve the Classical Return Mapping Schemes of the Infinitesimal Theory," *Computer Methods in Applied Mechanics and Engineering*, Vol. 99, 61-112.

Simo, J.C., Marsden, J.E. and Krishnaprasad, P.S., 1988 "The Hamiltonian Structure of Nonlinear Elasticity: The Material and Convective Representations of Solids, Rods and Plates," *Archive for Rational Mechanics*, Vol. 104, 125-183.

Simo, J.C. and Miehe, C., 1992 "Associative Coupled Thermoplasticity at Finite Strains: Formulation, Numerical Analysis and Implementation," *Computer Methods in Applied Mechanics and Engineering*, Vol. 98, 41-104.

Simo, J.C. and Taylor, R.L., 1985 "Consistent Tangent Operators for Rate-Independent Elastoplasticity," *Computer Methods in Applied Mechanics and Engineering*, Vol. 48, 101-118.

Simo, J.C., Taylor, R.L. and Wriggers, P., 1991 "A Note on Finite Element Implementation of Pressure Boundary Loading," *Communications in Applied Numerical Methods*, Vol. 50, 163-180.

Sjaardema, G.D., 1989 *NUMBERS: A Collection of Utilities for Pre- and Post-Processing Two- and Three-Dimensional EXODUS Finite Element Models*, SAND88-0737, Sandia National Laboratories, Albuquerque, NM.

Sjaardema, G.D., 1992 *GJOIN: A Program for Merging Two or More GENESIS Databases Version 1.4*, SAND92-2290, Sandia National Laboratories, Albuquerque, NM.

Sjaardema, G.D., 1993 *GENSHELL: A GENESIS Database Shell Transformation Program*, Sandia National Laboratories, Albuquerque, NM.

Sjaardema, G.D., 1993 *Overview of the Sandia National Laboratories Engineering Analysis Code Access System*, SAND92-2292, Sandia National Laboratories, Albuquerque, NM.

Sjaardema, G.D. and Krieg, R.D., 1987 *A Constitutive Model for the Consolidation of WIPP Crushed Salt and Its Use in Analyses of Backfilled Shaft and Drift Configurations*, SAND87-1977*UC-70, Sandia National Laboratories, Albuquerque, NM.

Stone, C.M., 1989 "Personal communication," Applied Mechanics Division, Sandia National Laboratories, Albuquerque, NM.

Stone, C.M., 1995 *SANTOS: A Two-Dimensional Finite Element Program faor the Quasistatic Large Deformation, Inelastic Response of Solids*, SAND90-0543, Sandia National Laboratories, Albuquerque, NM.

Stone, C.M., Krieg, R.D. and Beisinger, Z.E., 1988 *SANCHO - A Finite Element Computer Program for the Quasistatic, Large Deformation, Inelastic Response of Two-Dimensional Solids*, SAND84-2618, Sandia National Laboratories, Albuquerque, NM.

Stone, C. M. and Wellman, G. W., 1993 "Implementation of Ductile Failure in PRONTO2D and PRONTO3D", Memo to Distribution, Sandia National Laboratories, Albuquerque, NM.

Stone, C.M., Wellman, G.W. and Krieg, R.D., 1990 *A Vectorized Elastic/Plastic Power Law Hardening Material Model Including Luders Strain*, SAND90-0153, Sandia National Laboratories, Albuquerque, NM.

Strang, G. and Fix, G.J., 1973 *An Analysis of the Finite Element Method*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Swegle, J.W., 1978 *TOODY IV - A Computer Program for Two-Dimensional Wave Propagation*, SAND78-0552, Sandia National Laboratories, Albuquerque, NM.

Swegle, J.W., 1992 "Search Algorithm," Memo to Distribution, Sandia National Laboratories, Albuquerque, NM.

Swegle, J.W., Attaway, S.W., Heinstein, M.W., Mello, F.J. and Hicks, D.L., 1993 *An Analysis of Smoothed Particle Hydrodynamics*, SAND93-2513, Sandia National Laboratoaries, Albuquerque, NM.

Swenson, D.V. and Taylor, L.M., 1983 "A Finite Element Model for the Analysis of Tailored Pulse Stimulation of Boreholes," *International Journal for Numerical and Analytical Methods in Geomechanics*, Vol. 7, pp. 469-484.

Takaki, T. and Bogue, D.C., 1975 "The Extensional and Failure Properties of Polymer Melts," Vol. 19, pp. 419-433.

Taylor, L.M. and Becker, E.B., 1983 "Some Computational Aspects of Large Deformation, Rate-Dependent Plasticity Problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 41, No. 3, pp. 251-277.

Taylor, L.M., Chen, E.P. and Kuszmaul, J.S., 1986 "Microcrack-Induced Damage Accumulation in Brittle Rock Under Dynamic Loading," *Computer Methods in Applied Mechanics and Engineering*, Vol. 55, No. 3, pp. 301-320.

Taylor, L.M., Flanagan, D.P. and Mills-Curran, W.C., 1986 *The GENESIS Finite Element Mesh File Format*, SAND86-0910, Sandia National Laboratories, Albuquerque, NM.

Taylor, L.M. and Flanagan, D.P., 1987 *PRONTO 2D: A Two Dimensional Transient Solid Dynamics Program*, SAND86-0594, Sandia National Laboratories, Albuquerque, NM.

Taylor, L.M. and Flanagan, D.P., 1989 *PRONTO 3D: A Three Dimensional Transient Solid Dynamics Program*, SAND 87-1912, Sandia National Laboratories, Albuquerque, NM.

Thompson, S.L., 1977 *CSQII - An Eulerian Finite Difference Program for Two-Dimensional Material Response - Part 1, Material Selections*, SAND77-1339, Sandia National Laboratories, Albuquerque, NM.

Thorne, B.J., 1990 *A Damage Model for Rock Fragmentation and Comparison of Calculations with Blasting Experiments in Granite*, SAND90-1389, Sandia National Laboratories, Albuquerque, NM.

Thorne, B.J. and Preece, D.S., 1989 "Personal communications," Geoenergy Technology Department, Sandia National Laboratories, Albuquerque, NM.

Thrun, R., Goertner, J.F. and Harris, G.S., 1993 *Underwater Explosion Bubble Collapse Against a Flat Plate*, Seneca Lake Test Series Data Report, NSWC Report, NSWCDD/TR-92/482.

Timoshenko, S. and MacCollough, 1958 ???

Timoshenko, S. and Goodier, J.N., 1970 *Theory of Elasticity*, 3rd ed., McGraw-Hill, New York.

Treloar, 1994 ???

Truesdell, C., 1966 *The Elements of Continnum Mechanics*, Springer Verlag, New York.

Truesdell, C., 1977 *A First Course in Rational Continuum Mechanics, Vol. 1, General Concepts*, Academic Press, Inc., New York, p. 162.

Truesdell, C. and Noll, W., 1965 "Non-Linear Field Theories", in the *Handbook of Physics* by Flugge, Springer-Verlag, Berlin.

Valanis, K. and Lnadel, R.F., 1967 "The strain energy function of a hyperelastic material in terms of the extension ratios," *Journal of Applied Physics*, Vol. 38.

Von Neumann, J. and Richtyer, R.D., 1950 "A Method for the Numerical Calculation of Hydrodynamic Shocks," *Journal of Applied Physics*, Vol. 21, p. 232.

Warren, T.L. and Forrestal, M.J., 1997 "Effects of strain hardening and strain-rate sensitivity on the penetration of aluminum targets with spherical-nosed rods", *International Journal of Solids Structures* (submitted).

Warren, T.L. and Tabbara, M.R., 1997 *Spherical Cavity-Expansion Forcing Function in PRONTO3D for Application to Penetration Problems*, SAND97-1174, Sandia National Laboratories, Albuquerque, NM.

Weaver, Jr., W. and Johnson, P. R., 1984 *Finite Elements for Structural Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey.

Wellman, G. W., 1993 "Investigation of Mesh Dependencies in Ductile Failure for Transient Dynamics", Memo to Distribution, Sandia National Laboratories, Albuquerque, NM.

Wen, Y., Hicks, D. L. and Swegle, J. W., 1994 *Stabilizing S.P.H. with Conservative Smoothing*, SAND94-1932, Sandia National Laboratories, Albuquerque, NM.

Whirley, R. G., Engelmann, B. E. and Hallquist, J. O., 1991 *DYNA3D Users Manual*, Lawrence Livermore Laboratory, Livermore, CA.

Whirley, R.G., Hallquist, J.O. and Goudreau, G.L., 1988 *An Assessment of Numerical Algorithms for Plane Stress and Shell Elastoplasticity on Supercomputers*, UCRL-99690, Lawrence Livermore National Laboratory, Livermore, CA.

Wilkins, M.L. and Guinam, M.W., 1973 "Impact of CYlinders on a Rigid Boundary", *Journal of Applied Physics*, Vol. 44.

Williams, Landell, and Ferry, 1955 ???

Wriggers, P. and Simo, J.C., 1985 "A Note on Tangent Stiffness for Fully Nonlinear Contact Problems," *Communications in Applied Numerical Methods*, Vol. 1, 199-203.

Wriggers, P., Vu Van, T. and Stein, E., 1990 "Finite Element Formulation of Large Deformation Impact-Contact Problems with Friction," *Computers and Structures*, Vol. 37, 319-331.

Zeuch, Holcomb, and Lauson, 19?? ???

Zhang, P. and Geers, T. L., 1993 "Excitation of a fluid-filled, submerged spherical shell by a transient acoustic wave," *Journal Acoustical Society of America*, Vol. 93, pp. 696-705.

Zhong, Z.H. and Nilsson, L., 1990 "A Contact Searching Algorithm for General 3D Contact-Impact Problems," *Computers and Structures*, Vol. 34, No. 2, pp. 327-335.

Distribution:

MS9018 Central Technical Files, 8940-2
MS0899 Technical Library, 4916 (2)
MS0619 Review & Approval Desk for DOE/OSTI, 12690 (2)